MovieLens Final Project

HarvardX Data Science Program Professional Certificate

Mauricio Patón

January, 2021

Contents

1 Final Report **1**
1.1 **Executive Summary** 1
1.2 **Introduction** 1
1.3 **Methodology: Data Exploration** 2
1.4 **Model Development and Model Results** 16
1.5 **Conclusions** 35

1 Final Report

1.1 Executive Summary

In this work, the MovieLens data set of 10 million observations is studied. Firstly, a proper data exploration is conducted to understand the properties of each one of the variables provided in the set.

Then, different linear models are constructed in order to minimize de RMSE between the expected and the predicted rating. Different linear models incorporate the biases in the ratings by the user, the movie, the genre and or the year.

Several options of linear models are proposed that meet the target set for this homework, achieving a RMSE value lower than the threshold number.

1.2 Introduction

The Capstone Course of the HarvardX Data Science Program Professional Certificate offered in edX requires the application of a recommendation system based on the previous knowledge acquired in previous courses.

A recommender system is a tool used by retailers and/or video providers such as Netflix, Amazon Prime, Spotify to suggest movies to watch, songs to hear or products to purchase. Further description of what a recommender system does and is used fore can be checked on Wikipedia. Since the Netflix challenge (2009), several recommendation methods have been successfully implemented.

Several packages in R have been developed, such as the *Recommender* and the *recosystem* package. These packages deal with sparse matrices, due to the nature of the matrices/dataframes produced. Each user rates few movies (or items) of the entire dataset. Therefore, we have a lot of 0's (or unrated items) and few items that are rated. Sparse matrices collect those 0 as empty values to favor the storage of data in computers.

A good recommendation system should provide a reasonably accurate recommendation to the end-user. In order to measure the accuracy of a recommendation, different metrics can be used, such as the Root Mean Squared Error (RMSE), the Mean Squared Error (MSE). In this work, the RMSE will be the metric assigned to measure how good (or not) the recommendation system is.

1.3 Methodology: Data Exploration

To begin with, the dataset provided containing 10 M of observations should be loaded. Therefore, the script provided in the edX Capstone Course was used for that purpose. Note that since it was required to use the data many times, to avoid downloading it again and again, it was decided to save it the variables in an RData file.

1.3.1 Preliminary data exploration

Once the data was loaded, the exploratory analysis was conducted. Before proceeding with the modeling of the data, the first step consists of understanding the data at hand. Therefore, the functions from the basic package **summary**, and **head** were executed.

```
# Data Exploration
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     292      5 838983421               Outbreak (1995)
## 4      1     316      5 838983392               Stargate (1994)
## 5      1     329      5 838983392 Star Trek: Generations (1994)
## 6      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 3  Action|Drama|Sci-Fi|Thriller
## 4       Action|Adventure|Sci-Fi
## 5 Action|Adventure|Drama|Sci-Fi
## 6       Children|Comedy|Fantasy
```

```
# Find how many users and movies are
n_users <- n_distinct(edx$userId)
n_movies <- n_distinct(edx$movieId)
```

An initial exploration reveals that there are **69878 users** and **10677 movies** in the dataset. There are 10 M of observations because one user rates more than one movie and also, a movie can be rated by several users. With this information, a huge matrix of users rating per movie could be built. However, most of the ratings of this matrix do not correspond a rating (one user can watch certain amount of movies only). Matrices with few ratings and a lot of zeros or NA values are known as sparse matrices. In R, we could also

store this information in this type of matrices. The RecommenderLab and the Recosystem package are based on these type of matrices.

For the problem set in this work, we are not going to use these types of matrices. In this work, the edx and validation data will be used to provide a recommendation.

1.3.2 Initial data conversion

The summary also reveals that the time is set as timestamp. Therefore, a conversion using the **lubridate** package is required in order to get meaningful time data such as the week, month or hour of the movie. This information can then later be used for evaluating potential biases.

```
# Add to the train set the time of the movie and also other time variables that could be of use
edx_time <- edx %>% mutate(movie_title = str_sub(title, 1, -8),
                      year_released = as.numeric(str_sub(title, -5, -2)),
                      time_rating = as_datetime(timestamp)) %>%
                      mutate(weekday = weekdays(time_rating), week = week(time_rating),
                      day = lubridate::day(time_rating), month = month(time_rating),
                      year = year(time_rating), hour = hour(time_rating),
                      delta_years_rating = year(time_rating) - year_released) %>%
                      select(-timestamp, -title)

# Apply the same for the validation set
validation_time <- validation %>% mutate(movie_title = str_sub(title, 1, -8),
                      year_released = as.numeric(str_sub(title, -5, -2)),
                      time_rating = as_datetime(timestamp)) %>%
                      mutate(weekday = weekdays(time_rating), week = week(time_rating),
                      day = lubridate::day(time_rating), month = month(time_rating),
                      year = year(time_rating), hour = hour(time_rating),
                      delta_years_rating = year(time_rating) - year_released) %>%
                      select(-timestamp, -title)
```

1.3.3 Data visualisation

Let's get some graphical representations of the data for the different elements seen in the dataset.

1.3.3.1 Number of ratings per user {#userRatings}

Firstly, let's start by observing how many ratings are given per each user. For that, we can build histogram.

```
#Group the data to see how many ratings are given by each user
user_distribution <- edx_time %>% group_by(userId) %>% summarise(n = n())

# Calculate the median to plot as reference
median_user_rating <- median(user_distribution$n)

# Plot the histogram of user distribution
user_distribution %>% ggplot(aes(n)) + geom_histogram(bins = 30,
                                                      color = "black",
                                                      fill = "steelblue") +
                      xlab("Number of ratings") + ylab("Number of users") +
                      geom_vline(xintercept = 62, col = "maroon", lwd = 1, lty = 2)
```

```
# Show the number of ratings of the top users and the bottom
highest_raters <- user_distribution %>% top_n(10) %>% arrange(desc(n))
lowest_raters <-  user_distribution %>% top_n(-10) %>% arrange(n)
```

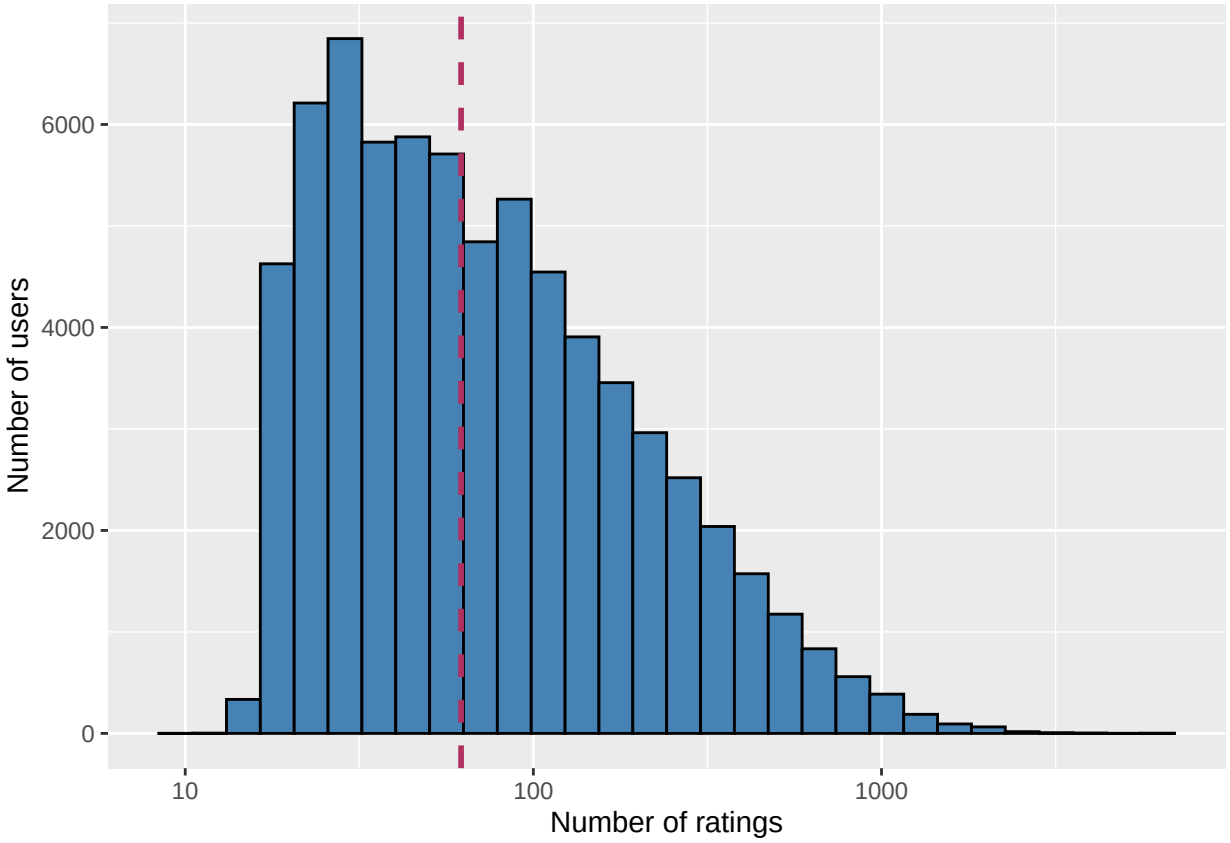

Figure 1: Histogram of number of movie ratings per user. Note that the x-axis is in logarithmic scale.

```
knitr::kable(highest_raters, caption = "Top-10 rating users") %>%
  kableExtra::kable_styling(latex_options = "hold_position")
```

Table 1: Top-10 rating users

userId	n
59269	6616
67385	6360
14463	4648
68259	4036
27468	4023
19635	3771
3817	3733
63134	3371
58357	3361
27584	3142

Figure 1 shows that the distribution for the number of ratings per user shows positive skewness. The user that has rated most movies has been the one rating 6616 movies, while the one that rated the least evaluated 10 movies. The difference appears to be very big, giving us an indication that possibly it would be better to provide a minimum number of ratings for the recommender algorithm. In addition, we might need to take into account that some users rate much more movies than others, and that might need somehow to be captured. We will explore this during the model development.

1.3.3.2 Data exploration: Number of ratings per movie

Analogously to the user ratings **??** user ratings, the same can be done for the number of ratings for each movie:

```
# Group dataset by movie
movie_distribution <- edx_time %>% group_by(movieId) %>% summarise(n = n())

movie_distribution %>%  ggplot(aes(n)) +
                        geom_histogram(bins = 30, color = "black", fill = "steelblue") +
                        scale_x_continuous(trans = "log10") +
                        xlab("Number of ratings") + ylab("Number of movies")
```

```
# Show the number of ratings of the top users and the bottom
most_rated_movies <- movie_distribution %>% top_n(10) %>% arrange(desc(n))
least_rated_movies <-  movie_distribution%>% top_n(-10) %>% arrange(n)
```

Figure 2 shows that few movies have been rated more than 10000 occasions. In addition, some number of movies have 10 ratings or less. This could also introduce a bias because the highest rated movies (or not) could be more favored than average or lowest rating ones. Again, we will revisit this during the model development.

1.3.3.3 Distribution of ratings

After seeing the distributions of users and movies respective to the ratings, it is time to observe how the rating values are distributed. Movies could receive a rating from 0.5 to 5, by increments of 0.5 stars. A histogram of the ratings can reveal which are the most frequent rates and if anything can be deducted from the ratings distribution.

```
# Calculate the average of the ratings
mu_rating <- mean(edx_time$rating)
```

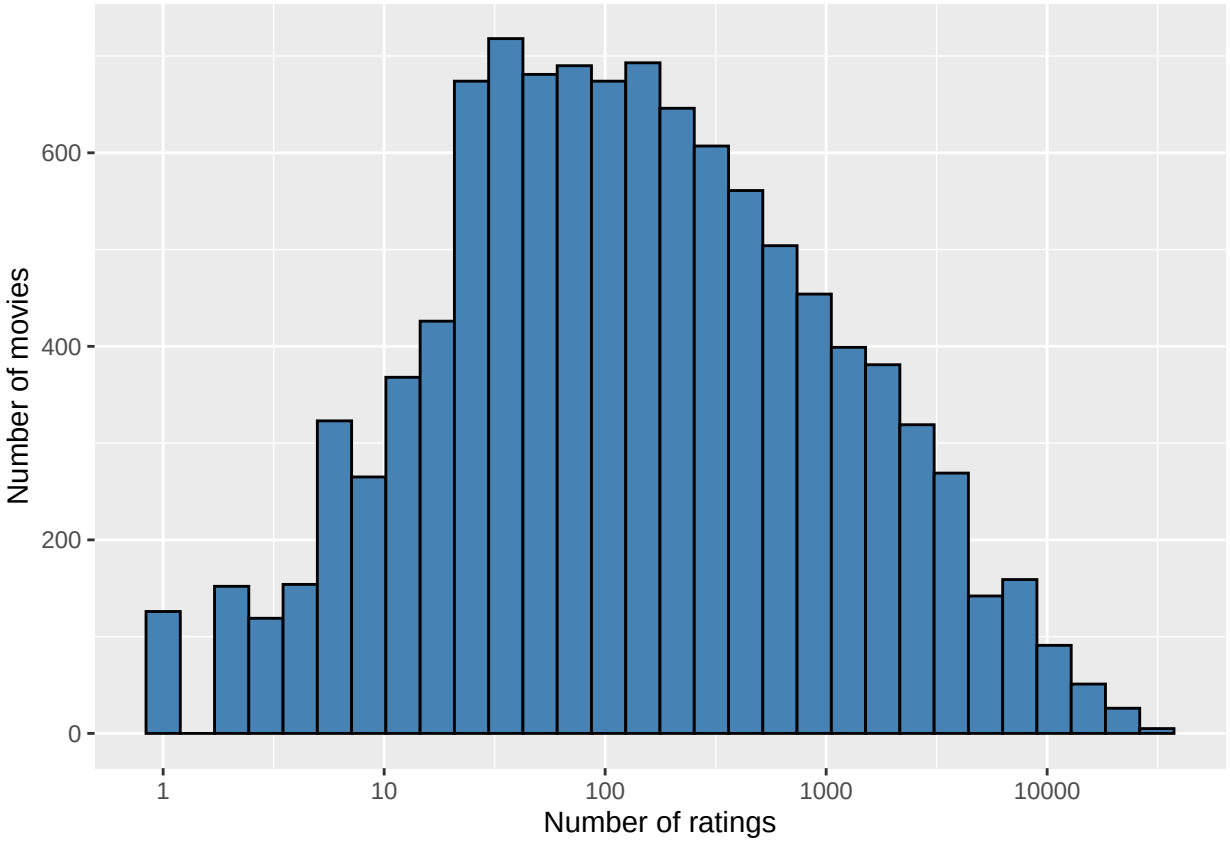

Figure 2: Histogram of number of ratings per movie. Please note that the x-axis is in logarithmic scale.

```r
# Histogram of the different ratings given
edx_time %>% ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.5, fill = "steelblue", col = "black") +
  scale_x_continuous(breaks = seq(0, 5, by = 0.5)) +
  scale_y_continuous(labels = number) +
  xlab("Movies Rating") +  ylab("Number of ratings") +
  # ggtitle("Histogram: Number of ratings for each rating") +
  geom_vline(xintercept = mu_rating, col = "maroon", lty = 2, lwd = 1)
```

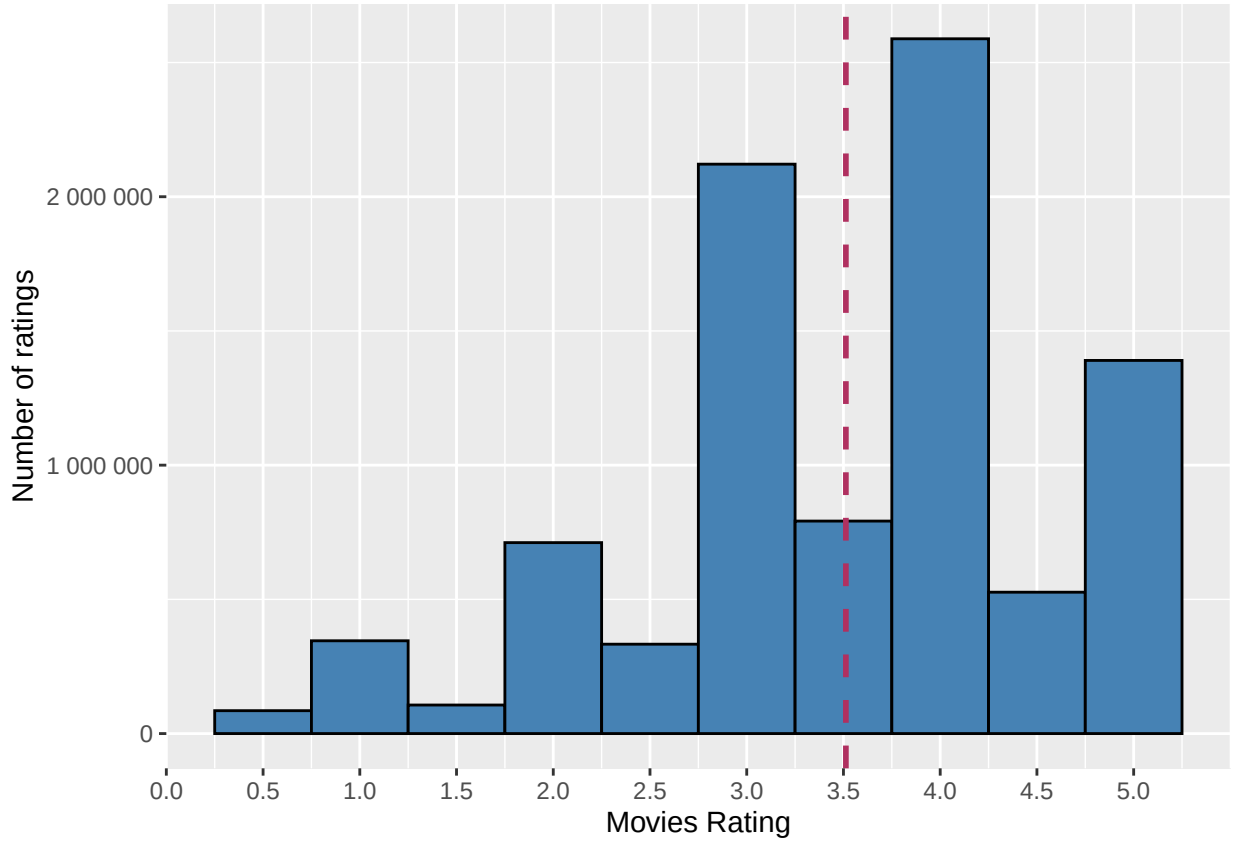

Figure 3: Histogram of number of ratings per movie ratings.

Figure 3 shows that most users are more prone to give full ratings than half ratings. One option could be to calculate the average of the users that only gave full ratings (to be considered). If we considered that only full marks or half-marks are given, a bell curve can be drawn between those points.

1.3.3.4 Distribution of genres

The dataset also includes some information regarding genres. One approach is to split the movies into genres for one line each. The drawback of this approach is that some movies belong to more than one genre and therefore, we might double count if we use the information in this way. Therefore we would need to probably create dummy columns for each genre and aggregate the data by movie again.

An alternative approach is to treat all defined genres as a different one. For simplicity purposes we have followed this approach in this work. We can see the distributions of number of ratings and rating values per genre.

```r
# First row corresponds to non genders listed, and therefore is removed
genres_exploration <- edx_time %>% group_by(genres) %>%
                                   summarise(n = n(), avg_rating = mean(rating),
                                             sd_rating = sd(rating)) %>%
                                   slice(-1)

#Averages of movies per genre
avg_movie_genre <- mean(genres_exploration$n)

# Histogram of number of ratings per genre
genres_exploration %>% ggplot(aes(n)) +
                       geom_histogram(bins = 30,
                                      fill = "steelblue",
                                      col = "black") +
                       scale_x_continuous(trans = "log10", labels = number) +
                       xlab("Number of genres") +
                       ylab("Number of movies") +
                       geom_vline(xintercept = avg_movie_genre,
                                  col = "maroon",
                                  lty = 2,
                                  lwd = 1)
```

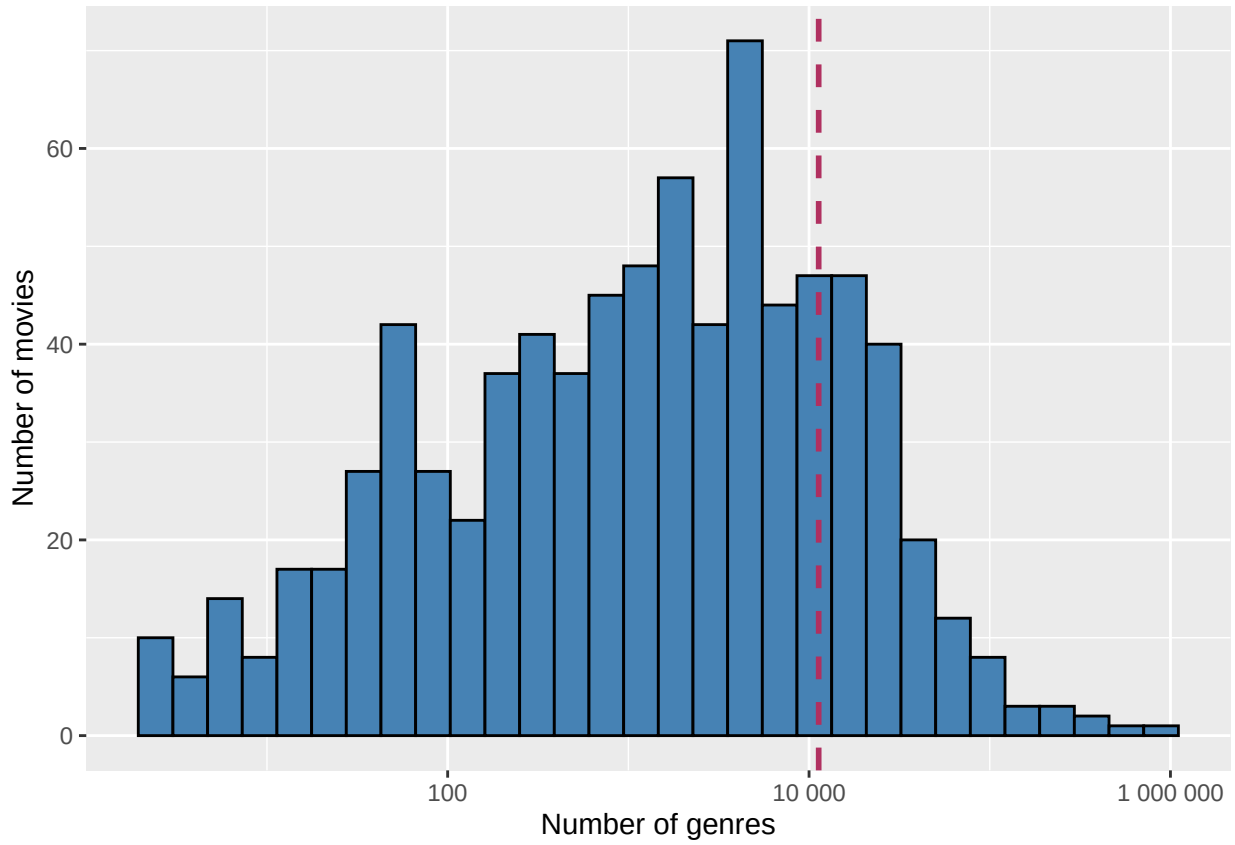

Figure 4: Number of movies per genre.

Figure 4 shows that some genres have much more movies than others, while other genres seem to be

under-represented.

Let's see now what occurs when we consider the average rating for each of the genres.

```
#Averages of movies per genre
avg_rating_genre <- mean(genres_exploration$avg_rating)

# Histogram of number of ratings per number of genres
genres_exploration %>% ggplot(aes(avg_rating)) +
                       geom_histogram(bins = 30,
                                      fill = "steelblue",
                                      col = "black") +
                       xlab("Rating") +
                       ylab("Number of genres") +
                       geom_vline(xintercept = avg_rating_genre,
                                  col = "maroon",
                                  lty = 2,
                                  lwd = 1)
```

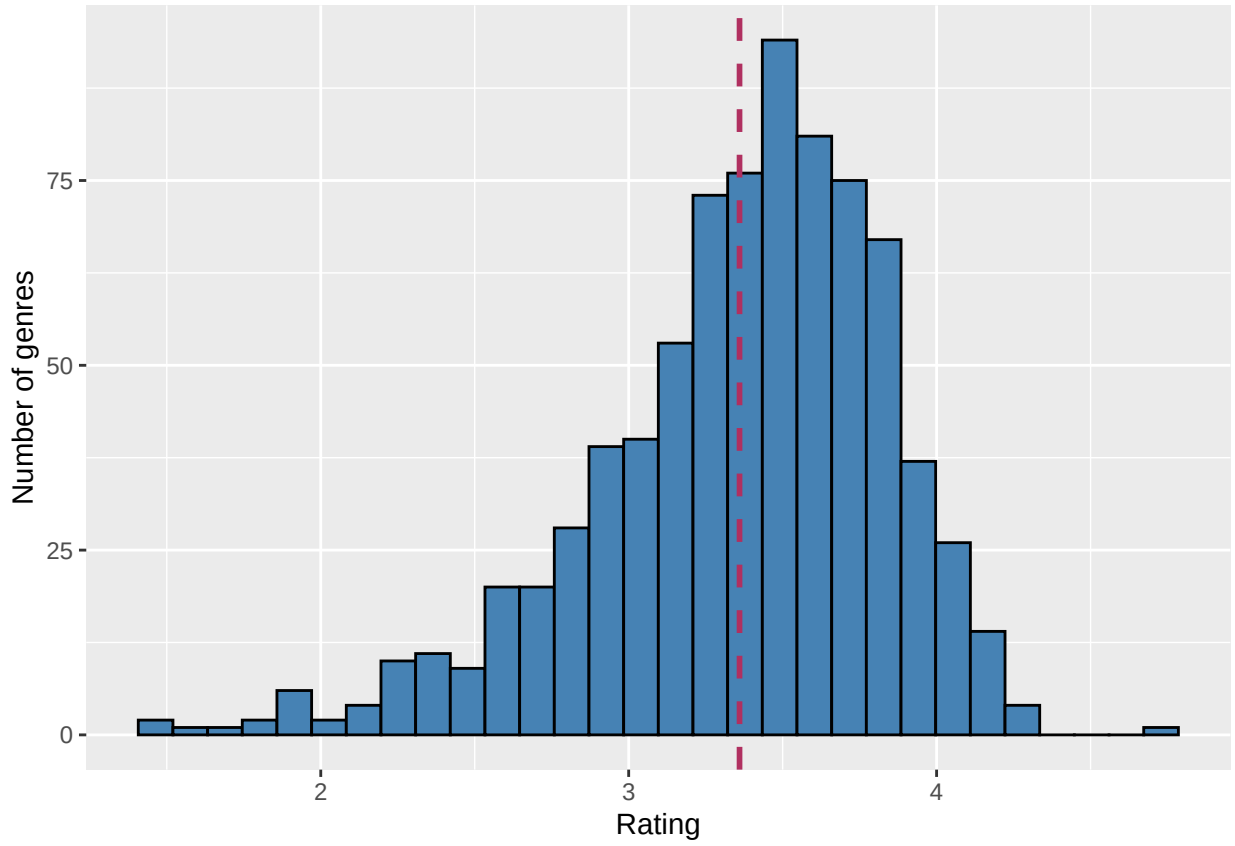

Figure 5: Average rating for each genre of movie rated.

The Figure 5 shows that, the ratings per genre fall under a negatively skewed curve. Some genres have very low values while some others are above 4. Most of the genres however fall under the average. This might indicate that the movies can be biased on the genre, but maybe the impact is not very big.

Let's plot the average rating for those genres that have more than 75000 ratings.

```r
# We might consider filter those that have more than 25, 50 or 100k ratings
min_rates <- 75000

genres_exploration %>% filter(n > min_rates) %>%
                       ggplot(aes(genres, avg_rating)) +
                       geom_bar(stat = "identity",
                                fill = "steelblue") +
                       geom_errorbar(aes(ymin = avg_rating - sd_rating,
                                         ymax = avg_rating + sd_rating)) +
                       geom_hline(yintercept = mu_rating, col = "maroon",
                                  lty = 2, lwd = 1) +
                       theme(axis.text.x = element_text(angle = 90))+
                       xlab("Genres") +
                       ylab("Average Rating")
```

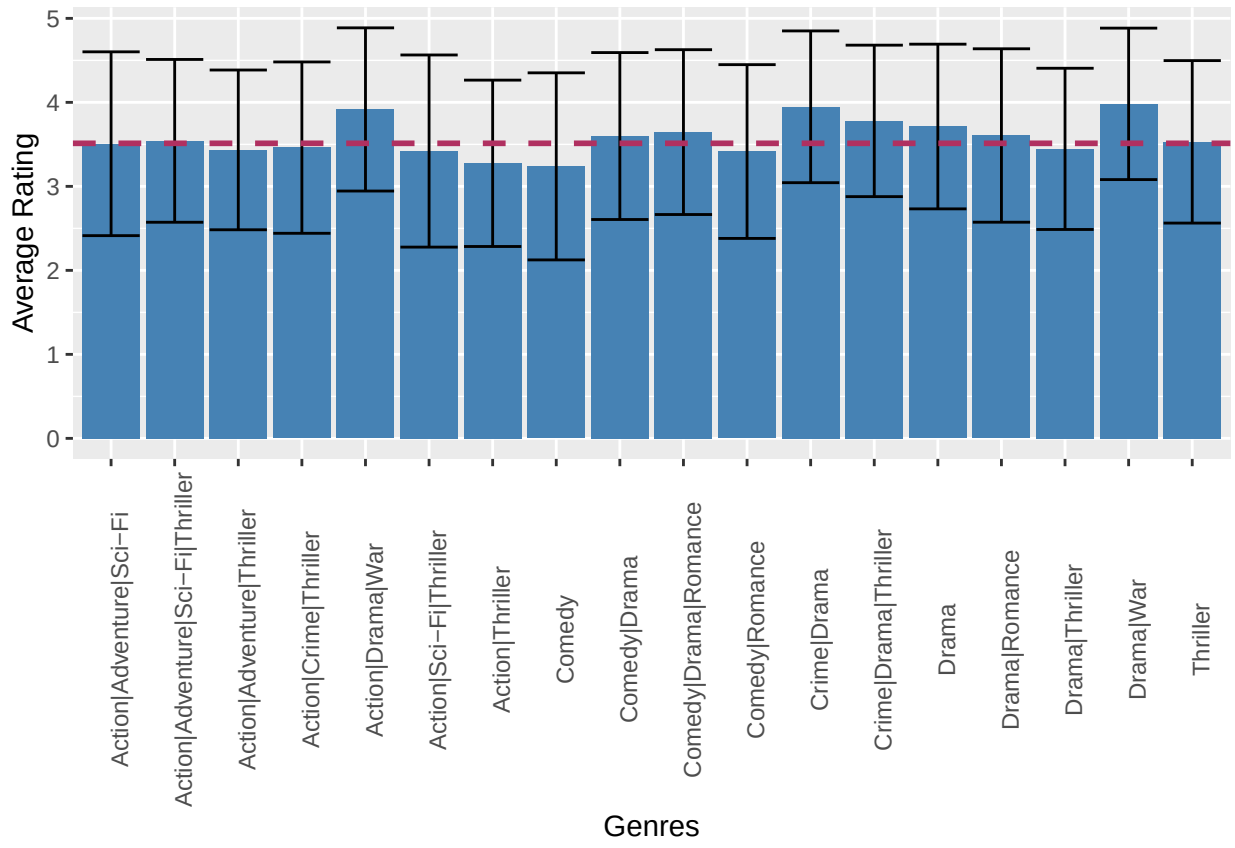

Figure 6: Average rating for each genre that has more than 75000 movies rated.

Figure 6 shows that, among those genres with more than 75000 movies, some genres appear to be more favored than other, with comedy being the one with the lowest rate and Drama|War being the highest rated. This indicates that possibly there is a bias in the genre that will have to be taken into account.

1.3.3.5 Date of ratings

So far, the ratings per movie and per user have been explored, along of the ratings per genre. One extra source of information available is the time. Both the release year of the movie, and also the time in which the

rating was conducted is information available that can be used to see if any there is any bias.

Let's see if there is any time effect either in the year it has been voted, the week of the year or the weekday.

```
# Plot of movies by year released
edx_time %>% group_by(year_released) %>%
             summarize(r = mean(rating), n = n()) %>%
             ggplot(aes(as.numeric(year_released), r)) +
                    geom_point() + geom_smooth(method = 'loess') +
                    scale_x_continuous(breaks = seq(1910,2010, by= 10)) +
                    xlab("Year of the movies released") +
                    ylab("Average Rating per year")
```

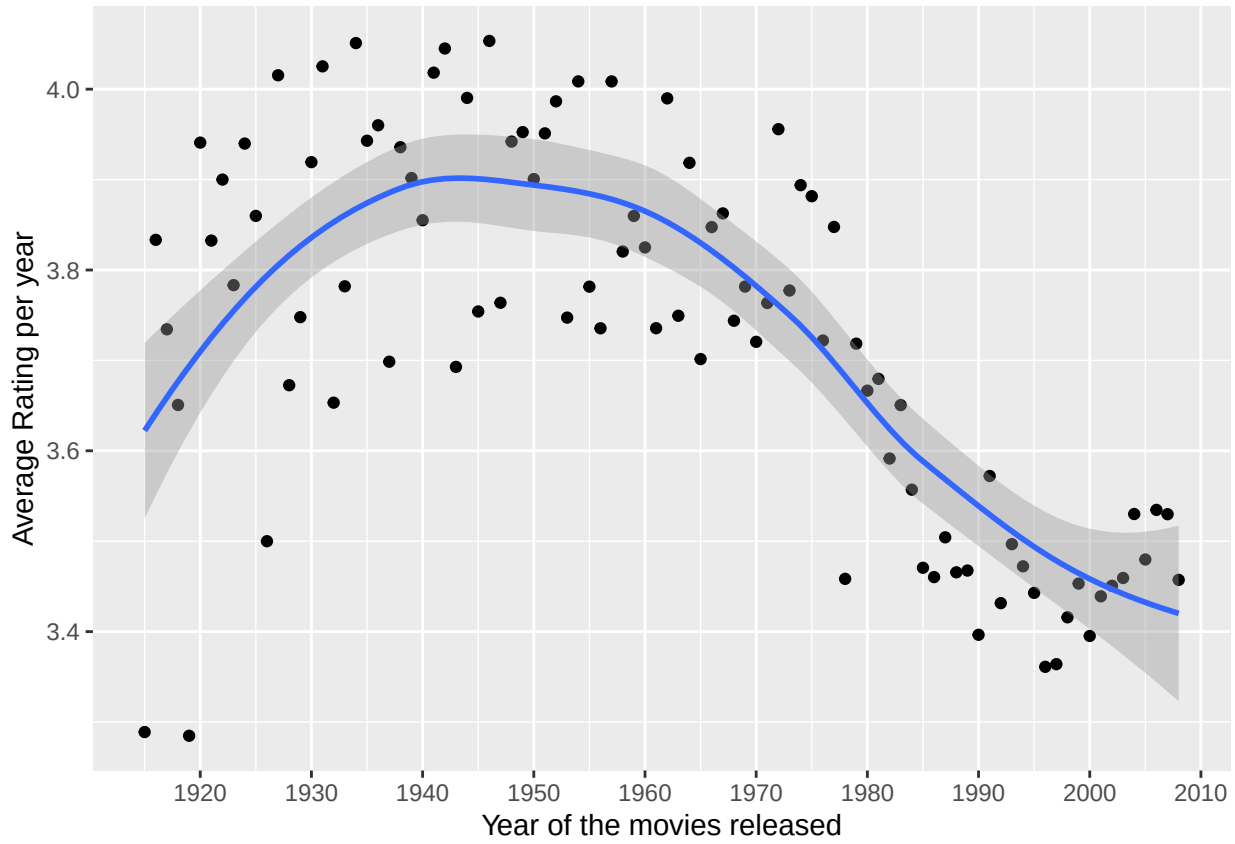

Figure 7: Average rating for movies released at each year.

Figure 7 shows that movies prior to 1975 have a higher average rating than movies after that year.

However, it needs to be seen if the average rating per year from the user side has changed or not. Let's plot the average rate for each movie rated each year.

```
# Plot of movies by year rated
edx_time %>% group_by(year) %>% summarize(r = mean(rating), n = n()) %>%
                    ggplot(aes(year, r)) +
                    geom_point() + geom_smooth(method = 'loess') +
                    scale_x_continuous(breaks = seq(1990,2010, by= 2)) +
                    xlab("Year of the movie rating") +
                    ylab("Average Rating per year")
```

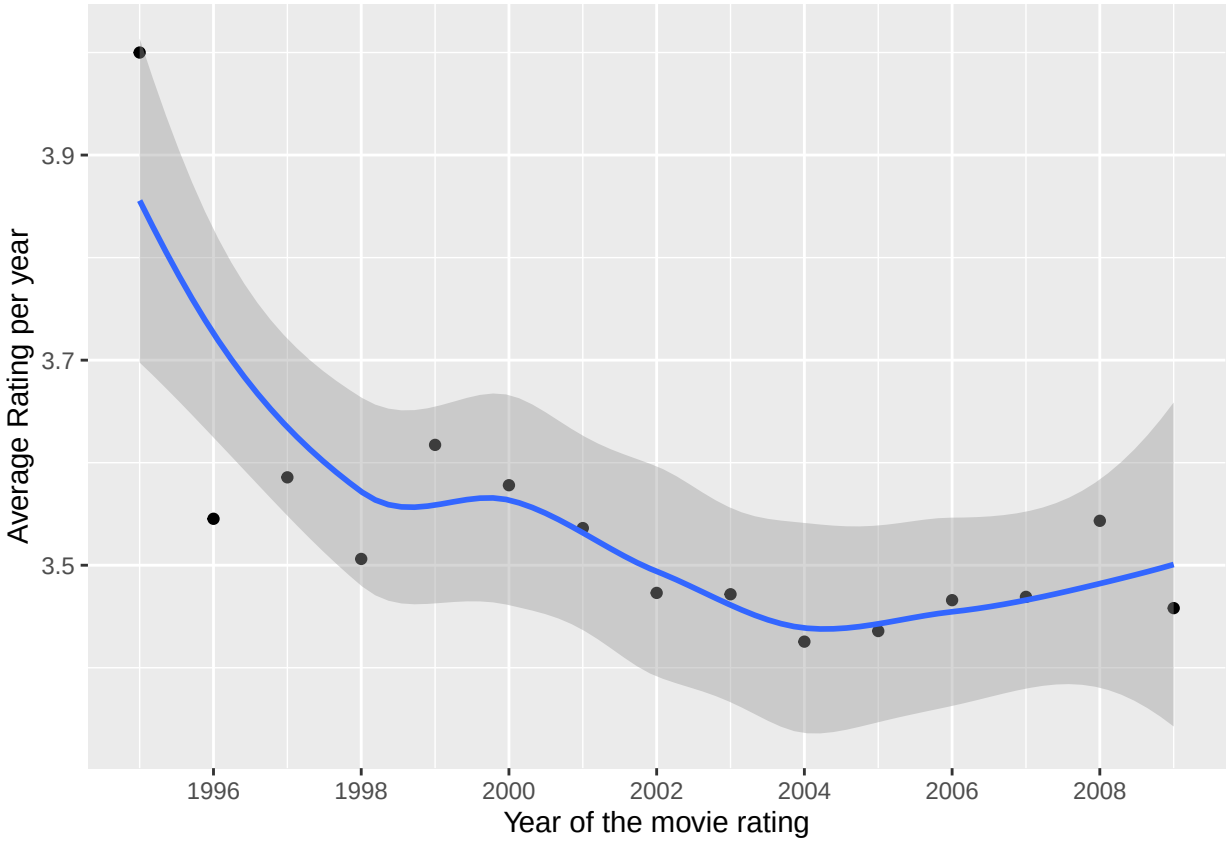

Figure 8: Average rating for movies rated at each year.

Figure 8 shows that from 1996, the average rate has stayed quite stable over the years. Therefore, the year in which the movie has been rated appears to have little impact.

Let's see the number of ratings over time to see which movies are rated more than others to see if there might be any influence by the amount of people rating.

```
# Plot of number of rates per year
edx_time %>% group_by(year_released) %>%
             summarise(r = mean(rating), n = n()) %>%
             ggplot(aes(as.numeric(year_released), n)) +
             geom_point() + geom_smooth(method = 'loess') +
             scale_x_continuous(breaks = seq(1910,2010, by= 10)) +
             scale_y_continuous(breaks = seq(0, 800000, by= 200000), labels = number) +
             # scale_y_continuous(trans = "log10") +
             xlab("Year of the movies released") +
             ylab("Number of ratings")
```

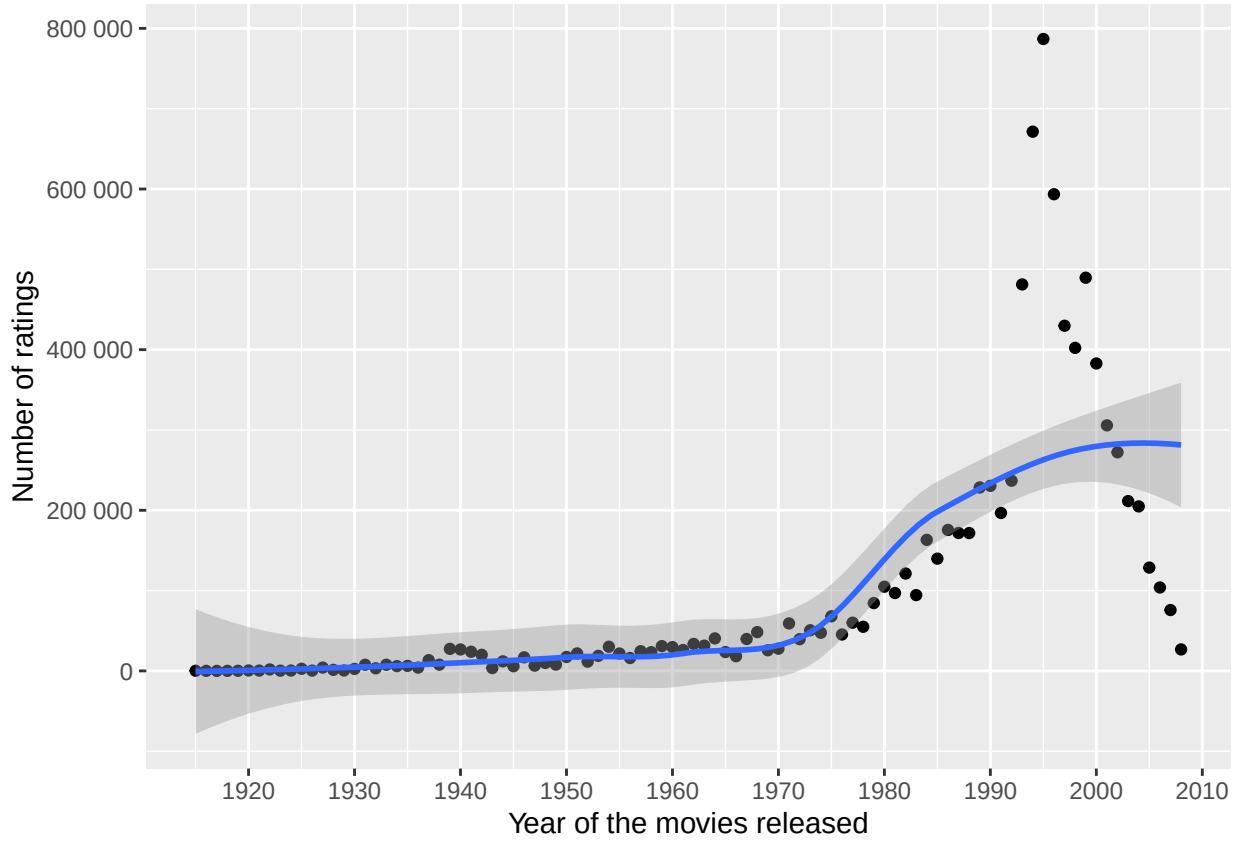

Figure 9: Number of rates per year

Figure 9 shows clearly that movies after 1980 are much more rated than the ones prior to that. Especially, it seems that from 1993, there is a spike in the rating of the movies that lats until early 2000's. Such difference in number of ratings could have an impact or a bias in the older movies, which have much less votes.

Let's also evaluate the average on the rates it in terms of the time difference (in years) between the year of release of the movie and the year of the ratings.

```
edx_time %>% group_by(delta_years_rating) %>%
             summarise(n = n(), mean = mean(rating)) %>%
             ggplot(aes(delta_years_rating,mean)) +
                    geom_point(col = "black", fill = "steelblue") +
                    geom_smooth(method = 'loess') +
                     xlab("Difference (in years) between movie released and its rate") +
                     ylab("Average rating")
```

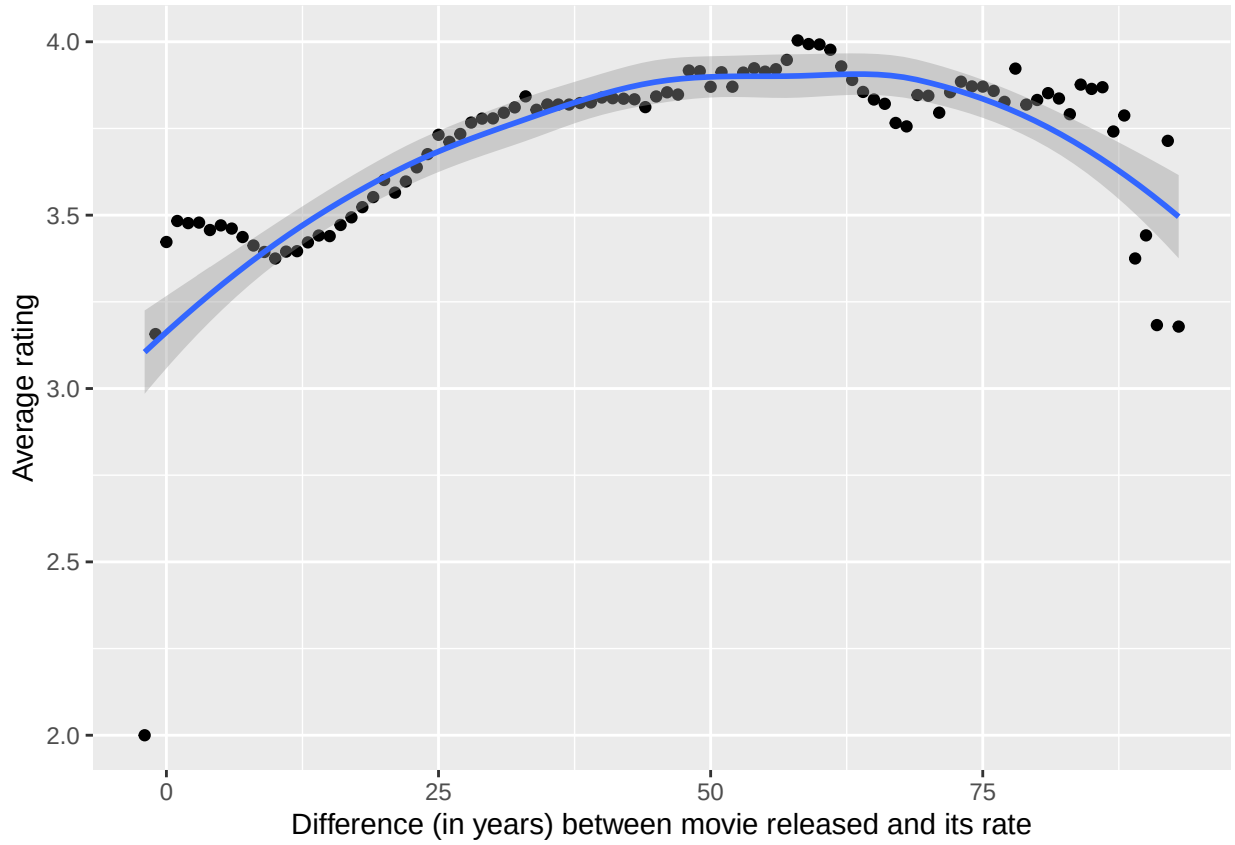

Figure 10: Average rating over time. Time is measured as the difference between the year released and the year the movie was rated.

Figure 9 shows that there might be some bias in the ratings, as older movies appear to have higher ratings than movies that have been rated with less time of difference. Therefore, it is possible that there is some bias.

```
# Check distribution of years between movie released and time
edx_time %>% ggplot(aes(delta_years_rating)) +
                    geom_histogram(col = "black", fill = "steelblue")
```

However, as we previously saw in Figure 11, older movies have much less ratings.

It seems that we might need to consider also time bias in our modeling approach, which is presented in the next section.

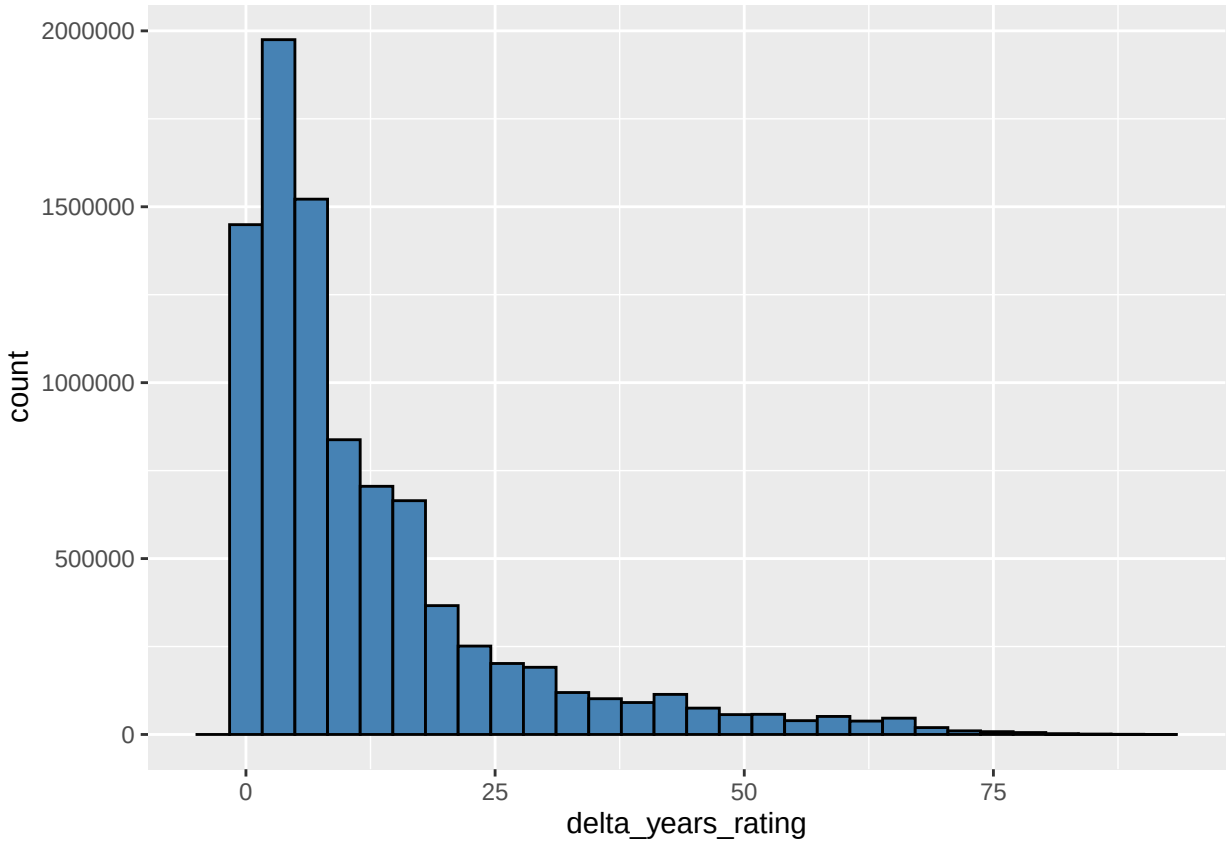

Figure 11: Average rating over time. Time is measured as the difference between the year released and the year the movie was rated.

1.4 Model Development and Model Results

As we have seen in the previous section, we need to consider biases for the following elements :

- User bias
- Movie bias
- Genre bias
- Time bias
- Number of ratings bias

In this section we will evaluate the impact of each of them alone and/or via a combination of them.

First of all, the requirement in this assignment is to obtain a RMSE value below a threshold of 0.86490. Therefore, the first step is to load the function RMSE which will calculate the error, along with the table that will record all of the results

```
#  Define RMSE as a function as it is going to be our measurement variable
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings-predicted_ratings)^2,na.rm=T))
}

#Minimum RMSE required
min_rmse <- 0.86490

# Initialize the tibble
rmse_results <- tibble()
```

1.4.1 Simplest approach: Use the average

The simplest approach for a prediction of the RMSE is to use just simply the average to estimate the ratings expected in the validation set

```
# Use the average as the estimator
just_avg_rmse <- RMSE(mu_rating, validation$rating)

# Collect the results
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Just the average",
                                 RMSE = just_avg_rmse,
                                 Meets.Goal = ifelse(just_avg_rmse < min_rmse,
                                                     "Yes", "No")))

# Print the table
knitr::kable(rmse_results,
             caption = "RMSE results for using the average.",
             digits = 5, align = c('l','c','c')) %>%
             kableExtra::kable_styling(latex_options = "hold_position")
```

Table 2: RMSE results for using the average.

Method	RMSE	Meets.Goal
Just the average	1.0612	No

Table 2 shows an RMSE for using the average of 1.0612018, which is quite high compared to the mean average of 3.5124652.

We would not be doing this exercise if the average was enough by itself. We need therefore to find of ways to improve the prediction.

1.4.2 Improve the model: Add Movie bias

Analogously to the Machine Learning Course in the HarvardX Data Science Program Professional Certificate of edX, we can incorporate the movie averages effects.

Let's compute first the movie bias

```
movie_avgs <- edx_time %>% group_by(movieId) %>%
                      summarise(b_m = mean(rating - mu_rating))

#Let's observe the distribution of the movie bias
movie_avgs %>% ggplot(aes(b_m)) +
               geom_histogram(binwidth = 0.4,
                              color = "black",
                              fill = "steelblue") +
               xlab('Movie bias') +
               ylab('Number of movies')
```

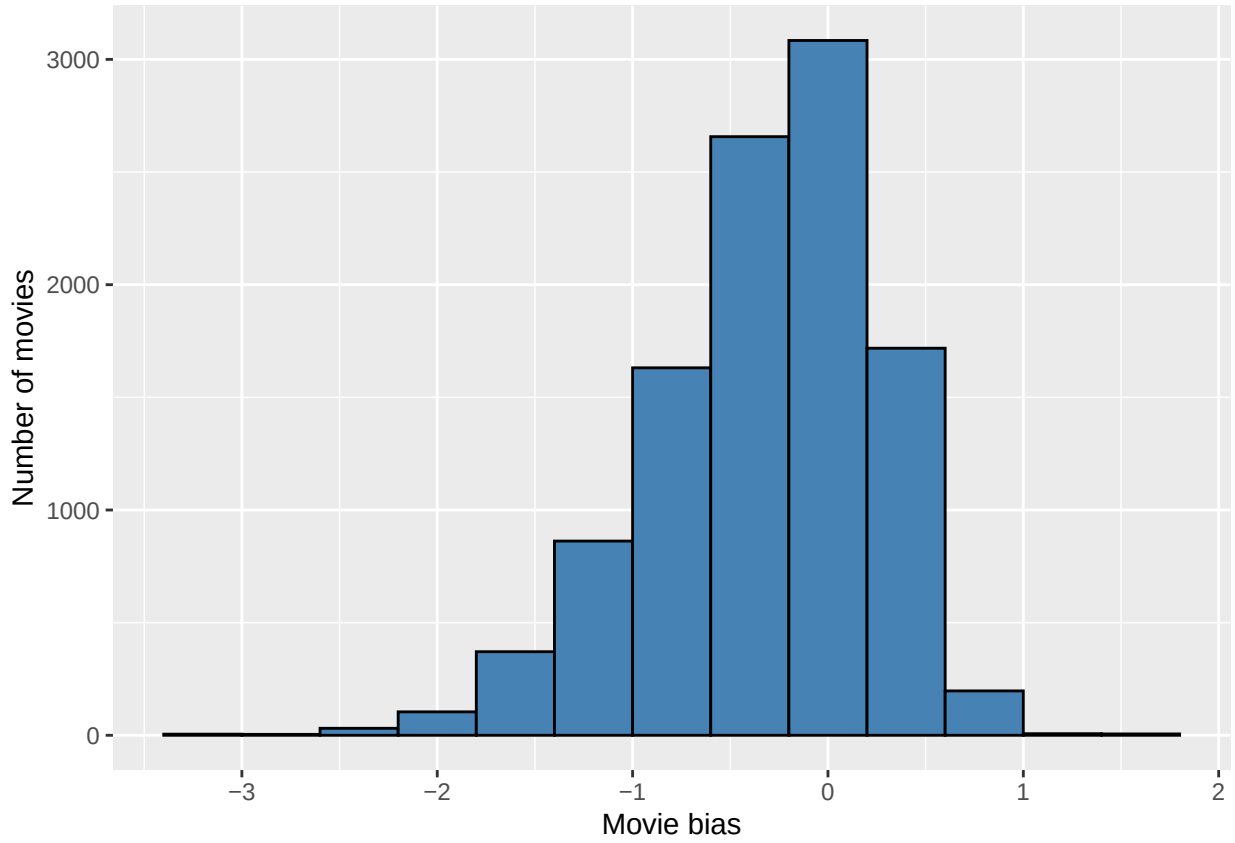

Figure 12: Histogram of the movie bias

Figure 12 shows skewness for the movie bias. Therefore, we will add it to the linear model using this code.

```
# To predict the ratings, we use mu_ratings (all movies)
# + the expected b_i for the validation set movies
movie_bias_ratings <- mu_rating + validation %>%
                      left_join(movie_avgs, by='movieId') %>%.$b_m

# Calculate the RMSE for the movie bias
model_movie_bias_rmse <- RMSE(movie_bias_ratings, validation$rating)

# Collect the results
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Movie Effect",
                                 RMSE = model_movie_bias_rmse,
                                 Meets.Goal = ifelse(model_movie_bias_rmse < min_rmse,
                                                     "Yes", "No")))
# Print the table
knitr::kable(rmse_results,
             caption = "RMSE results for
                        the average and the movie bias.",
             digits = 5, align = c('l','c','c')) %>%
             kableExtra::kable_styling(latex_options = "hold_position")
```

Table 3: RMSE results for the average and the movie bias.

Method	RMSE	Meets.Goal
Just the average	1.06120	No
Movie Effect	0.94391	No

Table 3 shows an RMSE for using the average of 0.9439087, which is an improvement but still above the target of 0.8649.

Therefore, we need to explore other biases to incorporate to our model in order to fit the prediction.

1.4.3 Improve the model: Add User bias

Another bias we observed was the user bias. Let's compute it in the same manner we did it for the movie bias.

```
# Compute user averages
user_avgs <- edx_time %>% group_by(userId) %>%
                      summarise(b_u = mean(rating - mu_rating))

#Let's observe the distribution of the user bias
user_avgs %>% ggplot(aes(b_u)) +
              geom_histogram(binwidth = 0.4,
                             color = "black",
                             fill = "steelblue") +
               xlab('User bias') +
               ylab('Number of movies')
```

Figure 15 Figure 13 shows skewness for the user bias alone. Therefore, we will add it to the linear model using this code.

```
# To predict the ratings, we use mu_ratings (all movies)
# + the expected b_i for the validation set movies
user_bias_ratings <- mu_rating + validation %>%
```

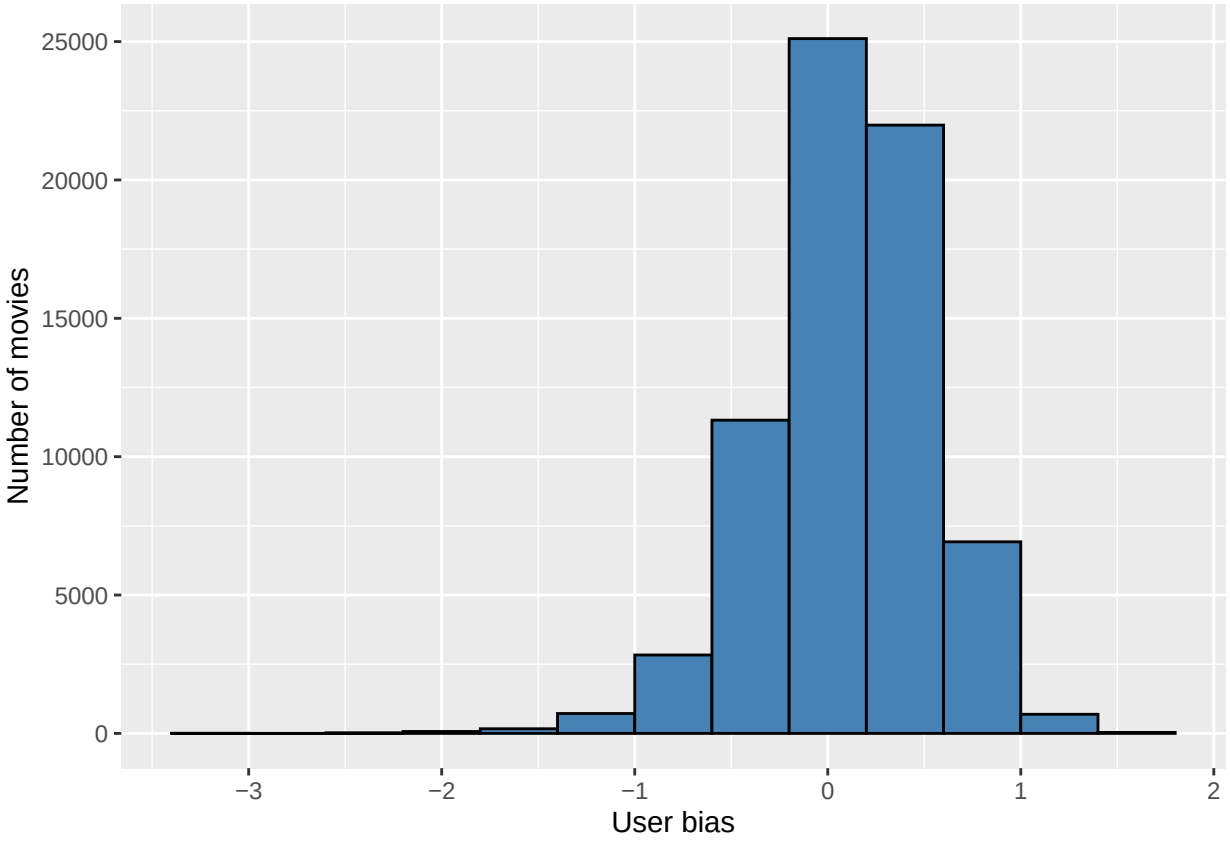

Figure 13: Histogram of the user bias only

```
                    left_join(user_avgs, by='userId') %>%.$b_u

model_user_bias_rmse <- RMSE(user_bias_ratings, validation$rating)


# Collect the results
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="User Effect",
                                     RMSE = model_user_bias_rmse,
                                     Meets.Goal = ifelse(model_user_bias_rmse<min_rmse,
                                                         "Yes", "No")))
# Print the table
knitr::kable(rmse_results,
             caption = "RMSE results for using
                        the average, the movie and
                        the user bias.",
             digits = 5, align = c('l','c','c')) %>%
             kableExtra::kable_styling(latex_options = "hold_position")
```

Table 4: RMSE results for using the average, the movie and the user bias.

Method	RMSE	Meets.Goal
Just the average	1.06120	No
Movie Effect	0.94391	No
User Effect	0.97834	No

Table 3 shows that using the users only bias appears to perform worse than considering only movies. What if we combine both ?

1.4.4 Improve the model: Add User and Movie bias

Let's see the histogram for **both** the movie and the genre bias.

```
user_movie_avgs <- edx_time %>% left_join(movie_avgs, by='movieId') %>%
                           group_by(userId) %>%
                           summarise(b_um = mean(rating - mu_rating - b_m))

#Let's observe the distribution of the user bias
user_movie_avgs %>% ggplot(aes(b_um)) +
                    geom_histogram(binwidth = 0.4,
                                   color = "black",
                                   fill = "steelblue") +
                xlab('Movie and user bias') +
                ylab('Number of movies')
```

Figure 14 shows a more centered distribution than the movie and the user bias alone. We will add it to the linear model using this code to evaluate if it performs better or not

```
# To predict the ratings, we use mu_ratings (all movies) and
# the expected b_i for the validation set movies
user_movie_bias_ratings <-  validation %>%
                            left_join(movie_avgs, by = 'movieId') %>%
                            left_join(user_movie_avgs, by = 'userId')  %>%
                            mutate(pred = mu_rating + b_um + b_m) %>% .$pred
```

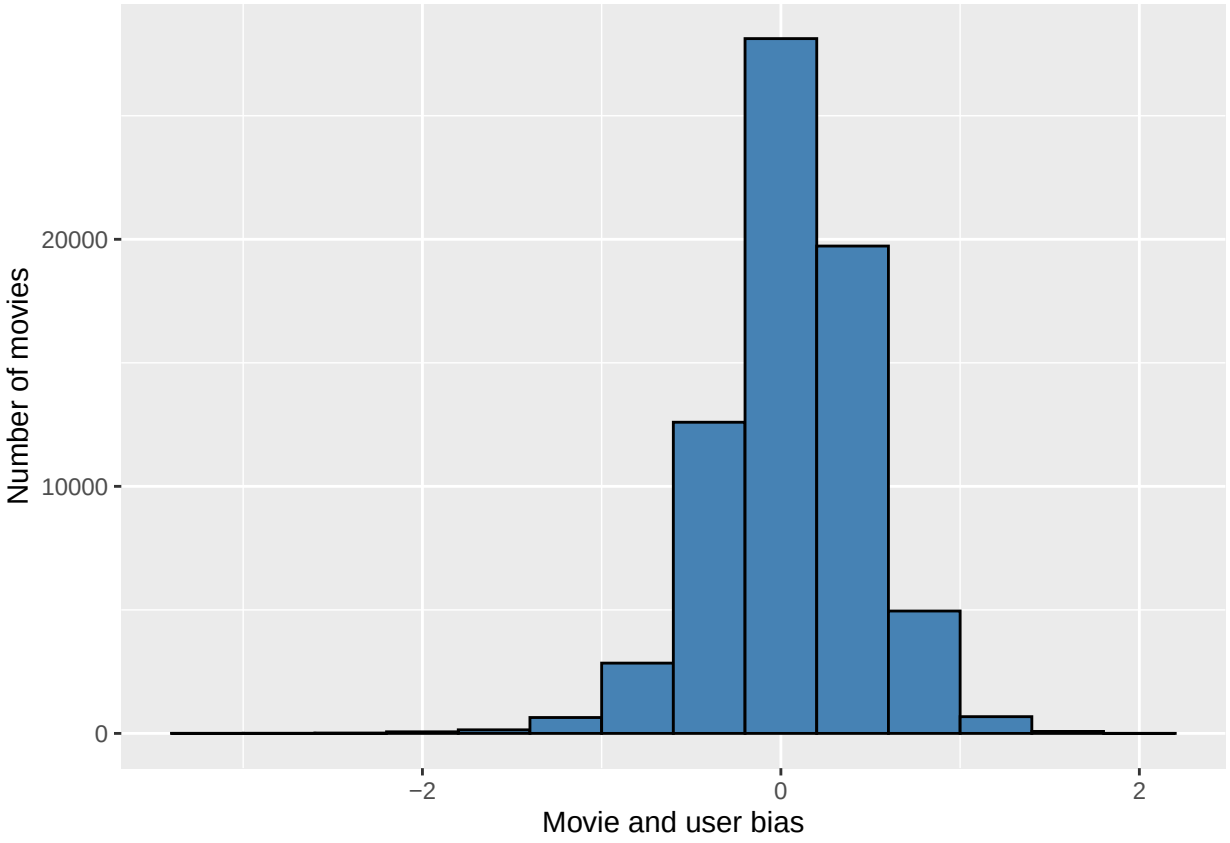

Figure 14: The distribution of the user and the movie bias

```
model_movie_user_bias_rmse <- RMSE(user_movie_bias_ratings, validation$rating)

# Collect the results
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Movie + User Effect",
                                 RMSE = model_movie_user_bias_rmse,
                                 Meets.Goal = ifelse(model_movie_user_bias_rmse<min_rmse,
                                                     "Yes", "No")))
# Print the table
knitr::kable(rmse_results,
             caption = "RMSE results for using
                        the average, the movie,
                        the user bias, and
                        the user + movie bias",
             digits = 5, align = c('l','c','c')) %>%
             kableExtra::kable_styling(latex_options = "hold_position")
```

Table 5: RMSE results for using the average, the movie, the user bias, and the user + movie bias

Method	RMSE	Meets.Goal
Just the average	1.06120	No
Movie Effect	0.94391	No
User Effect	0.97834	No
Movie + User Effect	0.86535	No

The RMSE obtained from the model 0.8653488 is still somehow much larger than the required of 0.8649. However it is an improvement from the previous ones when only one bias was considered.

Until here, we have only reproduced the model proposed in the machine learning course so far. Let's keep adding possible biases to see if the predictions of the model turn out better or not.

1.4.5 Genre bias

As we observed in the exploration of the data, there was the possibility of some bias in the genre. Let's compute the genre bias:

```
genre_avgs <- edx_time %>% left_join(movie_avgs, by = 'movieId') %>%
                      left_join(user_movie_avgs,  by = 'userId') %>%
                      group_by(genres) %>%
                      summarise(b_g = mean(rating - mu_rating - b_m - b_um))

#Let's observe the distribution of the user bias
genre_avgs %>% ggplot(aes(b_g)) +
               geom_histogram(bins = 30,
                              color = "black",
                              fill = "steelblue") +
               xlab('Movie + User + Genre bias') +
               ylab('Number of movies')
```

Figure 15 shows a centered distribution with thinner tails than the movie and the user bias previously showed. Since the user bias alone did not improve the prediction, we will show here all of them combined using this code to evaluate if it performs better or not.

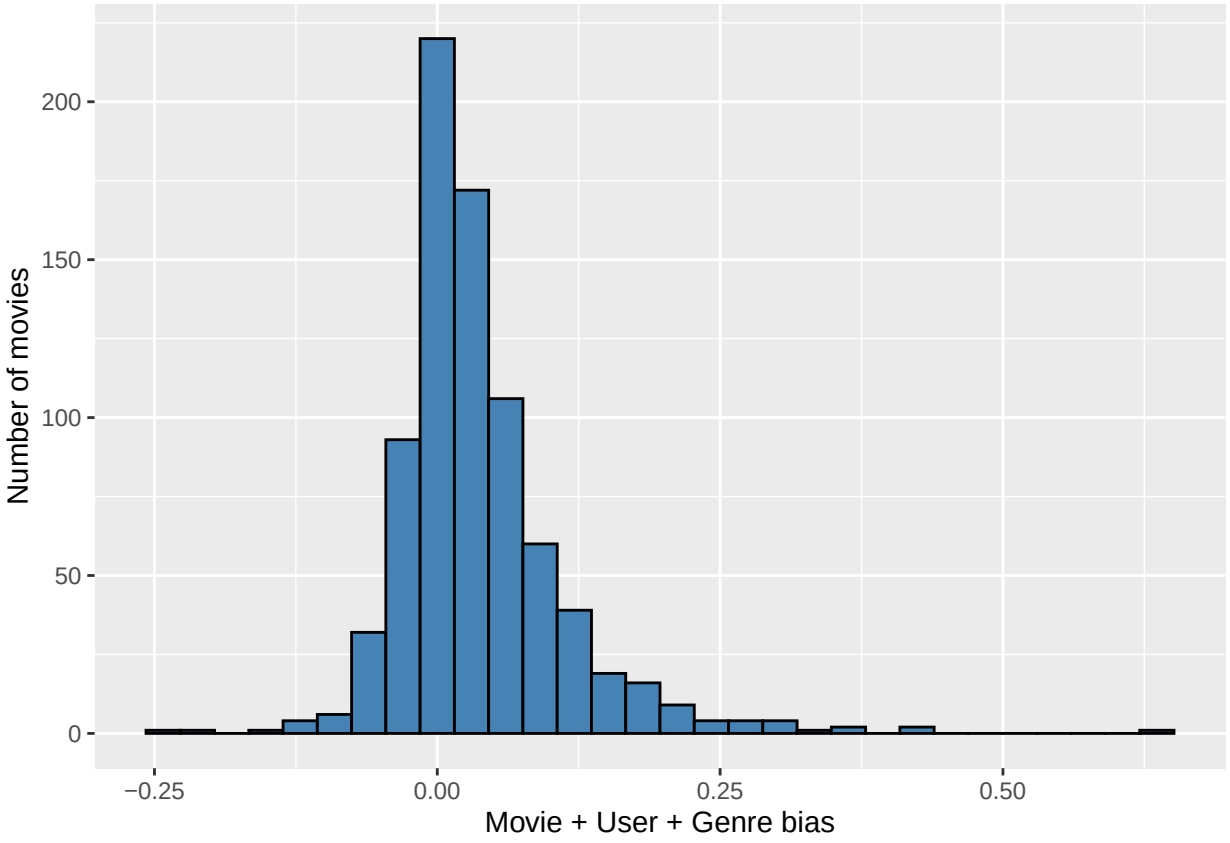

Figure 15: The distribution of the user, the movie bias **and the genre bias**

```
# To predict the ratings, we use mu_ratings (all movies)
# + the expected b_i for the validation set movies
genre_bias_ratings <- validation %>%
                          left_join(movie_avgs, by = 'movieId') %>%
                          left_join(user_movie_avgs,  by = 'userId')  %>%
                          left_join(genre_avgs, by = 'genres')  %>%
                          mutate(pred = mu_rating + b_um + b_m + b_g)  %>% .$pred

model_genre_bias_rmse <- RMSE(genre_bias_ratings, validation$rating)

# Collect the results
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Movie + User + Genre Effect",
                                 RMSE = model_genre_bias_rmse,
                                 Meets.Goal = ifelse(model_genre_bias_rmse<min_rmse,
                                                     "Yes", "No")))
# Print the table
knitr::kable(rmse_results,
             caption = "RMSE results for using
                        the average, the movie,
                        the user bias,
                        the user + movie bias and
                        the user + movie + genre bias",
             digits = 5, align = c('l','c','c')) %>%
             kableExtra::kable_styling(latex_options = "hold_position")
```

Table 6: RMSE results for using the average, the movie, the user bias, the user + movie bias and the user + movie + genre bias

Method	RMSE	Meets.Goal
Just the average	1.06120	No
Movie Effect	0.94391	No
User Effect	0.97834	No
Movie + User Effect	0.86535	No
Movie + User + Genre Effect	0.86495	No

The genre bias provides a small improvement in our predictions. However, we are still above the required threshold.

Next step, we will add the time bias in our computations.

1.4.6 Time bias

As we observed in the exploration of the data, there was the possibility of some bias also in terms of time. We will approach this via two possible biases, the year in which the movie was released and also in the difference between the rating of the movie and the year of the movie.

1.4.6.1 Time bias: Year Release The time bias considering the release of the movie can be evaluated with the following histogram in Figure 16:

```
time_avgs <- edx_time  %>% left_join(movie_avgs, by = 'movieId') %>%
                          left_join(user_movie_avgs,  by = 'userId')  %>%
                          left_join(genre_avgs, by = 'genres') %>%
                          group_by(year_released) %>%
```

```
                        summarise(b_yr = mean(rating - mu_rating - b_m -
                                              b_um - b_g))

#Let's observe the distribution of the user bias
time_avgs %>% ggplot(aes(b_yr)) +
              geom_histogram(bins = 30,
                             color = "black",
                             fill = "steelblue") +
               xlab('Movie + User + Genre + Year Released bias') +
               ylab('Number of movies')
```

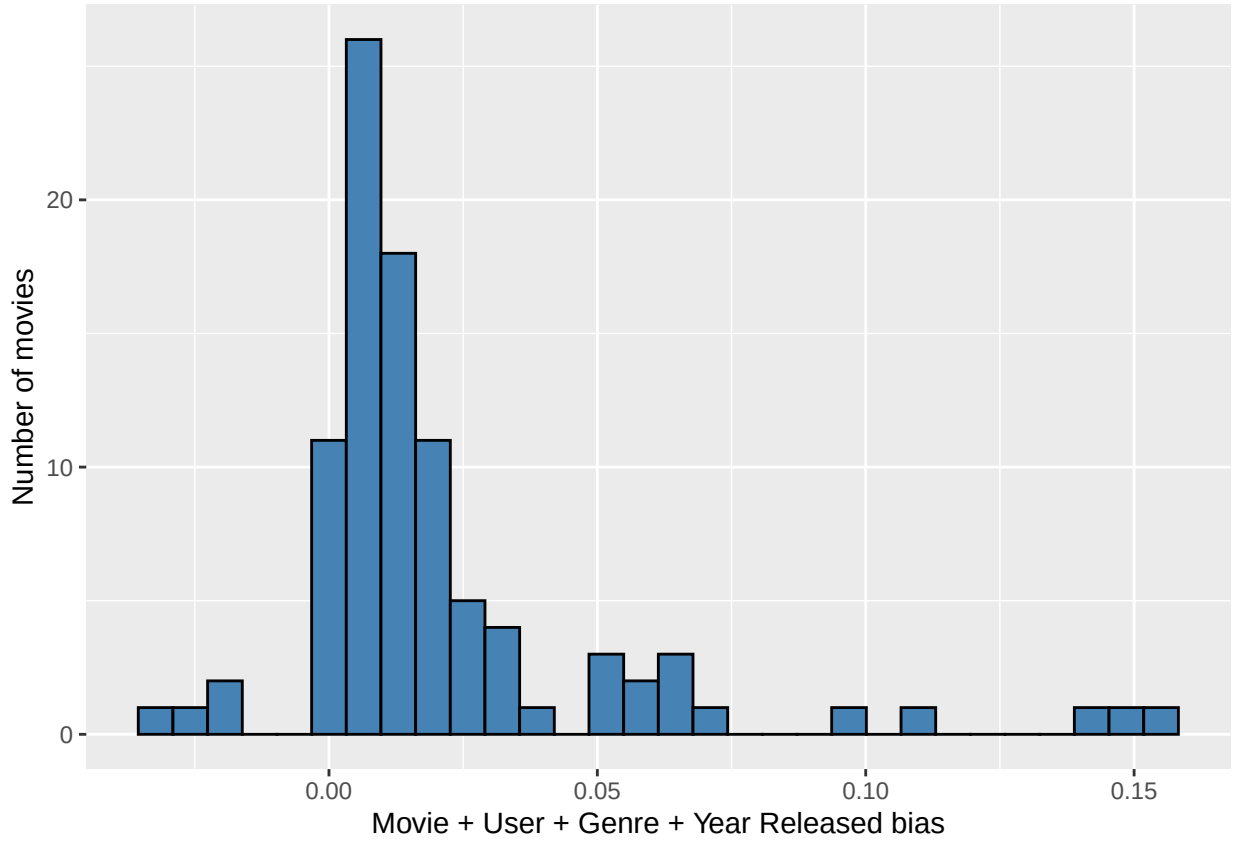

Figure 16: The distribution of the user, the movie bias, the genre bias **and the time of the release of the movie bias**

The bias show in Figure 16 appears centered to the 0 but with some odd tails particularly in the right handside. Let's see how the effect of the incorporation of this bias into the model.

```
# To predict the ratings, we use mu_ratings (all movies)
# + the expected b_i for the validation set movies
year_released_bias_ratings <- validation_time %>%
                                left_join(movie_avgs, by = 'movieId') %>%
                                left_join(user_movie_avgs,  by = 'userId') %>%
                                left_join(genre_avgs, by = 'genres') %>%
                                left_join(time_avgs, by = 'year_released')  %>%
                                mutate(pred = mu_rating + b_um + b_m + b_g + b_yr) %>%
```

```
                                    .$pred

model_year_bias_rmse <- RMSE(year_released_bias_ratings, validation_time$rating)

# Collect the results
rmse_results <- bind_rows(rmse_results,
                          tibble(Method="Movie + User + Genre + Year Released Effect",
                                 RMSE = model_year_bias_rmse,
                                 Meets.Goal = ifelse(model_year_bias_rmse<min_rmse,
                                                     "Yes", "No")))
# Print the table
knitr::kable(rmse_results,
             caption = "RMSE results for using
                        the average, the movie,
                        the user bias,
                        the user + movie bias and
                        the user + movie + genre bias",
             digits = 5, align = c('l','c','c')) %>%
             kableExtra::kable_styling(latex_options = "hold_position")
```

Table 7: RMSE results for using the average, the movie, the user bias, the user + movie bias and the user + movie + genre bias

Method	RMSE	Meets.Goal
Just the average	1.06120	No
Movie Effect	0.94391	No
User Effect	0.97834	No
Movie + User Effect	0.86535	No
Movie + User + Genre Effect	0.86495	No
Movie + User + Genre + Year Released Effect	0.86476	Yes

Despite the odd shape of the distribution for the bias, the incorporation of the time bias with the year of release of the movie es the first approach that gives us a RMSE of 0.8647606. This value below is the required threshold.

1.4.7 Time bias: Year Rating

Since we consider that the year released bias has an odd shape, we decided to evaluate the effect of time by the difference in time between the rating and the movie released. We can see the bias distribution in the histogram in Figure 17.

```
time_rating_avgs <- edx_time  %>% left_join(movie_avgs, by = 'movieId') %>%
                                  left_join(user_movie_avgs, by = 'userId') %>%
                                  left_join(genre_avgs, by = 'genres') %>%
                                  group_by(delta_years_rating) %>%
                                  summarise(b_y = mean(rating - mu_rating - b_m -
                                                       b_um - b_g))

#Let's observe the distribution of the user bias
time_rating_avgs %>% ggplot(aes(b_y)) +
                     geom_histogram(bins = 30,
                                    color = "black",
                                    fill = "steelblue") +
```

```
            xlab('Movie + User + Genre bias + Year Rating bias') +
            ylab('Number of movies')
```

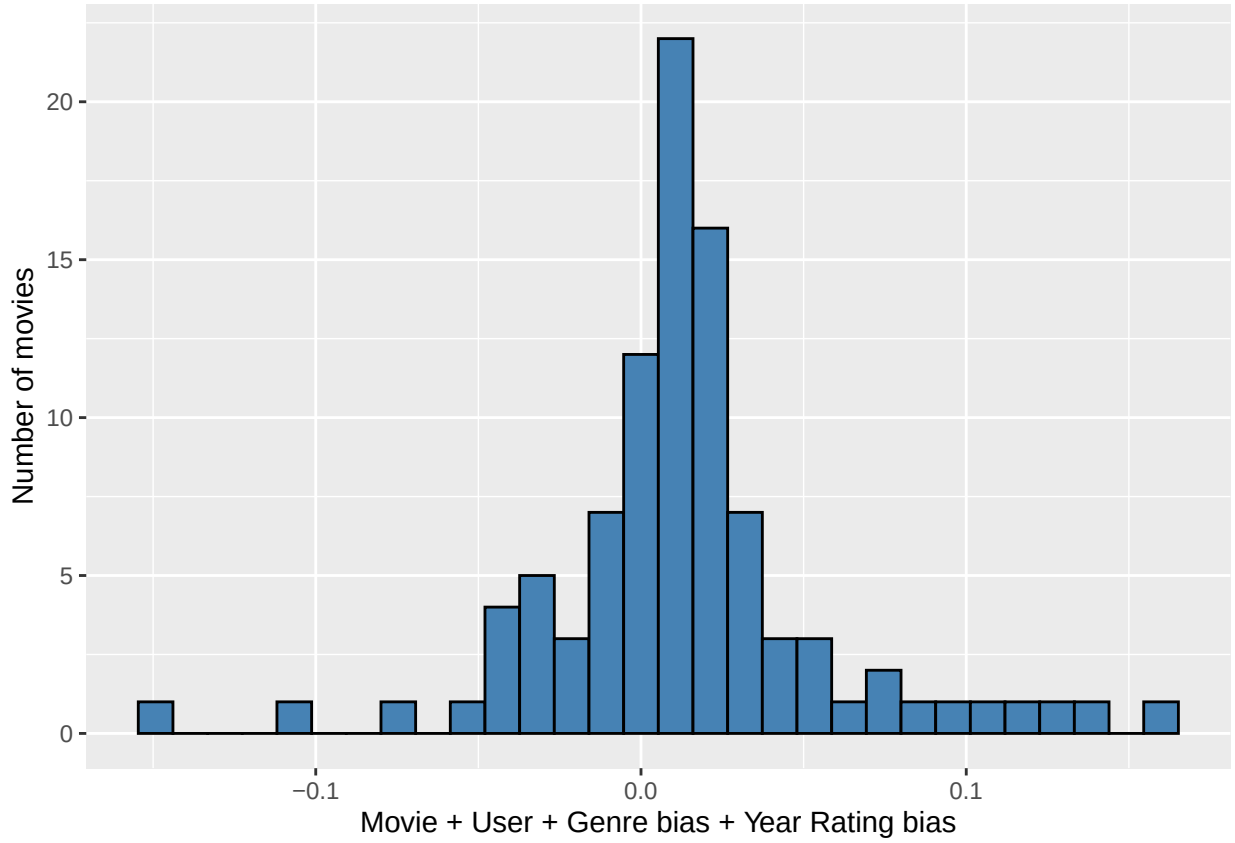

Figure 17: The distribution of the user, the movie bias, the genre bias **and the difference between the year of rating and the movie released bias**

The time bias appears more centered and somehow more continuous than the bias observed when considering the year released alone (16). Now it is time to evaluate the behavior of the linear model when adding this bias.

```
# To predict the ratings, we use mu_ratings (all movies)
#+ the expected b_i for the validation set movies
year_rating_bias_ratings <- validation_time %>% mutate(delta_years_rating = year -
                                                       year_released) %>%
                                  left_join(movie_avgs, by = 'movieId') %>%
                                  left_join(user_movie_avgs,  by = 'userId') %>%
                                  left_join(genre_avgs, by = 'genres') %>%
                                  left_join(time_rating_avgs, by = 'delta_years_rating')  %>%
                                  mutate(pred = mu_rating + b_um + b_m + b_g + b_y)  %>%
                                  .$pred

model_year_rating_bias_rmse <- RMSE(year_rating_bias_ratings, validation_time$rating)

# Collect the results
rmse_results <- bind_rows(rmse_results,
```

```
                          tibble(Method = "Movie + User + Genre + Year Rating Effect",
                                 RMSE = model_year_rating_bias_rmse,
                                 Meets.Goal = ifelse(model_year_rating_bias_rmse<min_rmse,
                                                     "Yes", "No")))
# Print the table
knitr::kable(rmse_results,
             caption = "RMSE results for using
                        the average, the movie,
                        the user bias,
                        the user + movie bias and
                        the user + movie + genre bias",
             digits = 5, align = c('l','c','c')) %>%
             kableExtra::kable_styling(latex_options = "hold_position")
```

Table 8: RMSE results for using the average, the movie, the user bias, the user + movie bias and the user + movie + genre bias

Method	RMSE	Meets.Goal
Just the average	1.06120	No
Movie Effect	0.94391	No
User Effect	0.97834	No
Movie + User Effect	0.86535	No
Movie + User + Genre Effect	0.86495	No
Movie + User + Genre + Year Released Effect	0.86476	Yes
Movie + User + Genre + Year Rating Effect	0.86454	Yes

The incorporation of the time bias via the approach of the difference between the year rated and the year of release of the movie gives us an RMSE of 0.86454. This value below the RMSE obtained when taking into account only the year in which the movie was released. Also, it falls the required threshold of 0.8649.

1.4.8 Regularisation bias

As we observed in Figure 1, some movies have been much more voted than others. This was also already shown in the Data Science Program, in the Machine Learning Course. Therefore, a penalty for those movies that have been watched most should be added. This penalty is added as a λ value.

1.4.8.1 Regularisation Movie and User Effect

To begin with, let's consider the simplest approach, which is the one considered in the course of Machine Learning. We regularise the movie and user effect, and we evaluate the regularisation with different values of λ fron 0 to 10.

```
#Declare function to be used to evaluate different lambdas (penalties)
movie_user_reg_rmse <- function(l){
  # Calculate the average of ratings
  mu_rating <- mean(edx$rating)
  # Calculate movie bias
  b_m <- edx_time %>%
    group_by(movieId) %>%
    summarize(b_m = sum(rating - mu_rating)/(n()+l))
  # Add user bias
  b_um <- edx_time %>%
    left_join(b_m, by="movieId") %>%
    group_by(userId) %>%
```

```
    summarize(b_um = sum(rating - b_m - mu_rating)/(n()+l))
  # Predict ratings for validation set
  predicted_ratings <- validation %>%
    left_join(b_m, by = "movieId") %>%
    left_join(b_um, by = "userId") %>%
    mutate(pred = mu_rating + b_um + b_m) %>%
    .$pred
  # Calculate and return RMSE
  return(RMSE(predicted_ratings, validation$rating))
}

# Let's evaluate with different values of lamba
lambdas <- seq(0, 10, 0.5)
rmses <- sapply(lambdas, movie_user_reg_rmse)

# Plot the RMSE for different lamdas
qplot(lambdas, rmses, xlab = 'lambda' , ylab = 'RMSE')
```

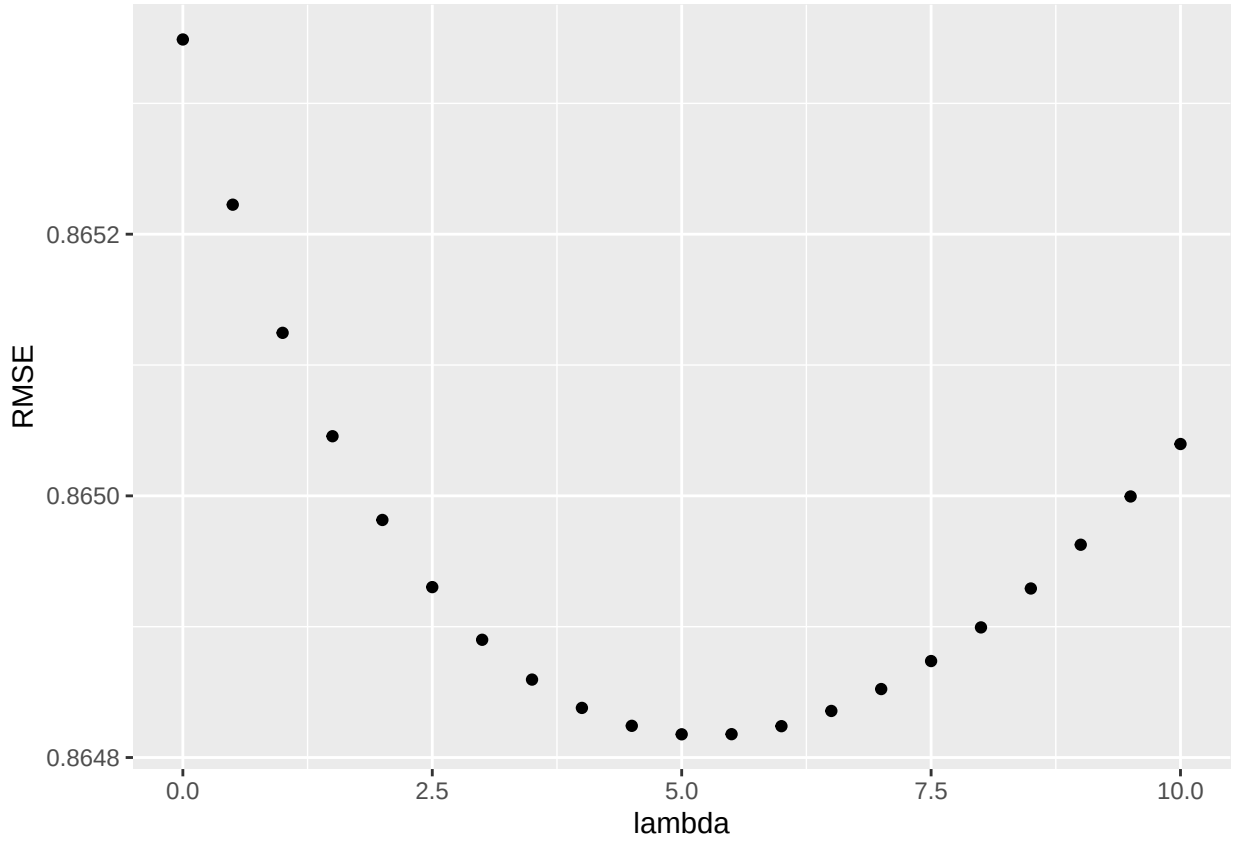

Figure 18: Regularisation plot for the movies with the user and movie effect accounting for the number of movies rated

```
lambda_min <- lambdas[which.min(rmses)]
```

Since the minimum RMSE is obtained with a $\lambda = 5$, we will run the regularised model with that value. The RMSE result obtained can be observed in the Table 9

```
# We re-calculate the model with the lambda that gives us the minimum RMSE
model_movie_user_reg_bias_rmse <- movie_user_reg_rmse(lambda_min)

# Collect the results
rmse_results <- bind_rows(rmse_results,
                          tibble(Method = "Movie + User Regularisation Effect",
                                 RMSE = model_movie_user_reg_bias_rmse,
                                 Meets.Goal = ifelse(model_movie_user_reg_bias_rmse<min_rmse,
                                                     "Yes", "No")))

# Print the table
knitr::kable(rmse_results,
             caption = "RMSE results for using
                        the average, the movie,
                        the user bias,
                        the user + movie bias and
                        the user + movie + genre bias",
             digits = 5, align = c('l','c','c')) %>%
             kableExtra::kable_styling(latex_options = "hold_position")
```

Table 9: RMSE results for using the average, the movie, the user bias, the user + movie bias and the user + movie + genre bias

Method	RMSE	Meets.Goal
Just the average	1.06120	No
Movie Effect	0.94391	No
User Effect	0.97834	No
Movie + User Effect	0.86535	No
Movie + User + Genre Effect	0.86495	No
Movie + User + Genre + Year Released Effect	0.86476	Yes
Movie + User + Genre + Year Rating Effect	0.86454	Yes
Movie + User Regularisation Effect	0.86482	Yes

We see in the table that with a $\lambda = 5$, the RMSE is below the threshold.

1.4.8.2 Regularisation Movie, User and Genre Effect

Next, we should incorporate the regularisation effect with the genre effect, which proved insufficient to be below the 0.8649 threshold value.

```
#Declare function to be used to evaluate different lambdas (penalties)
movie_user_genre_reg_rmse <- function(l){
  # Calculate the average of ratings
  mu_rating <- mean(edx$rating)
  # Calculate movie bias
    b_m <- edx_time %>%
    group_by(movieId) %>%
    summarize(b_m = sum(rating - mu_rating) / (n()+l))
  # Calculate user bias
  b_um <- edx_time %>%
    left_join(b_m, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_um = sum(rating - b_m - mu_rating) / (n()+l))
  # Calculate genre bias
```

```
  b_g <- edx_time %>%
    left_join(b_m,  by = 'movieId') %>%
    left_join(b_um, by = 'userId') %>%
    group_by(genres) %>%
    summarise(b_g = sum(rating  - b_m - b_um- mu_rating) / (n() + l))
  # Predict ratings for validation set
  predicted_ratings <- validation %>%
    left_join(b_m, by = "movieId") %>%
    left_join(b_um, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    mutate(pred = mu_rating + b_um + b_m + b_g) %>%
    .$pred
  # Calculate and return RMSE
  return(RMSE(predicted_ratings, validation$rating))
}

# Analogously to the movie and user effect, we calculate different values of lambda
lambdas <- seq(0, 10, 0.5)
rmses_g <- sapply(lambdas, movie_user_genre_reg_rmse)

qplot(lambdas, rmses_g)
```

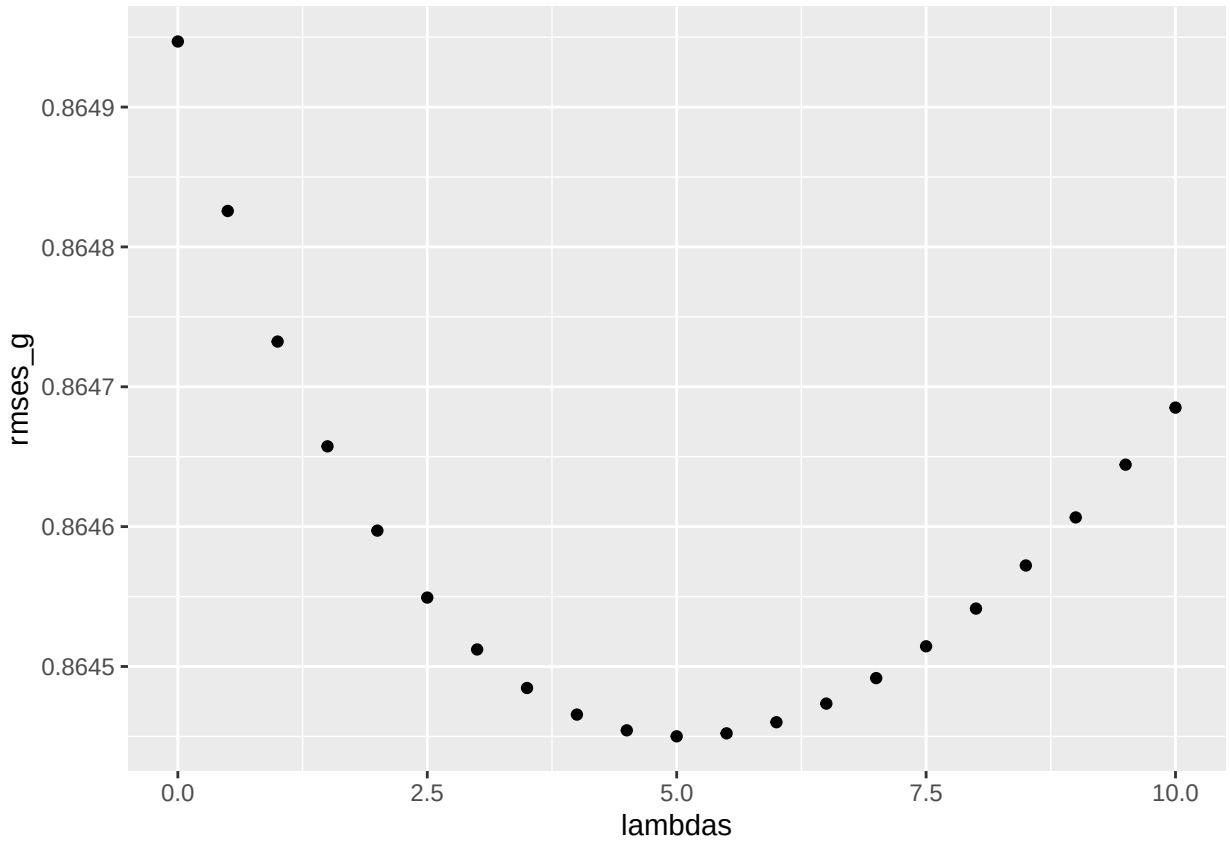

Figure 19: Regularisation plot for the movies with the user and movie effect accounting for the number of movies rated

```
lambda_min <- lambdas[which.min(rmses_g)]
```

Since the minimum RMSE is obtained with a $\lambda = 5$, we will run the regularised model with that value. The RMSE result obtained can be observed in the Table 10

```
# We re-calculate the model with the lambda that gives us the minimum RMSE
model_movie_user_gen_reg_bias_rmse <- movie_user_genre_reg_rmse(lambda_min)

# Collect the results
rmse_results <- bind_rows(rmse_results,
                          tibble(Method = "Movie + User +
                                          Genre Regularisation Effect",
                                 RMSE = model_movie_user_gen_reg_bias_rmse,
                                 Meets.Goal = ifelse(model_movie_user_gen_reg_bias_rmse<min_rmse,
                                                     "Yes", "No")))

# Print the table
knitr::kable(rmse_results,
             caption = "RMSE results for using
                        the average, the movie,
                        the user bias,
                        the user + movie bias and
                        the user + movie + genre bias",
             digits = 5, align = c('l','c','c')) %>%
             kableExtra::kable_styling(latex_options = "hold_position")
```

Table 10: RMSE results for using the average, the movie, the user bias, the user + movie bias and the user + movie + genre bias

Method	RMSE	Meets.Goal
Just the average	1.06120	No
Movie Effect	0.94391	No
User Effect	0.97834	No
Movie + User Effect	0.86535	No
Movie + User + Genre Effect	0.86495	No
Movie + User + Genre + Year Released Effect	0.86476	Yes
Movie + User + Genre + Year Rating Effect	0.86454	Yes
Movie + User Regularisation Effect	0.86482	Yes
Movie + User + Genre Regularisation Effect	0.86445	Yes

1.4.8.3 Regularisation Movie, User, Genre and Year Effect

Next, we should incorporate the regularisation effect with the year effect. We will consider in this case only the best approach for the year effect, which consisted in considering the difference in years between the rating and the year in which the movie was released. Adding the year bias with the genre, movie and user effect proved sufficient to be below the 0.8649 threshold value. However, let's see how good can be our linear model prediction by adding the regularisation bias

```
#Declare function to be used to evaluate different lambdas (penalties)
movie_user_genre_year_reg_rmse <- function(l){
  # Calculate the average of ratings
  mu_rating <- mean(edx$rating)
  # Calculate movie bias
    b_m <- edx_time %>%
```

```
    group_by(movieId) %>%
    summarize(b_m = sum(rating - mu_rating) / (n()+l))
  # Calculate user bias
  b_um <- edx_time %>%
    left_join(b_m, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_um = sum(rating - b_m - mu_rating) / (n()+l))
  # Calculate genre bias
  b_g <- edx_time %>%
    left_join(b_m,  by = 'movieId') %>%
    left_join(b_um, by = 'userId') %>%
    group_by(genres) %>%
    summarise(b_g = sum(rating  - b_m - b_um- mu_rating) / (n() + l))
  # Calculate year bias (as delta of time between year rated and year released)
  b_yr <- edx_time  %>%
    left_join(movie_avgs, by = 'movieId') %>%
    left_join(user_movie_avgs,  by = 'userId') %>%
    left_join(genre_avgs, by = 'genres') %>%
    group_by(delta_years_rating) %>%
    summarise(b_yr = sum(rating - mu_rating - b_m - b_um - b_g) / (n() + l))
  # Predict ratings for validation set
  predicted_ratings <- validation_time %>%
    left_join(b_m, by = "movieId") %>%
    left_join(b_um, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_yr, by = "delta_years_rating") %>%
    mutate(pred = mu_rating + b_um + b_m + b_g + b_yr) %>%
    .$pred
  # Calculate and return RMSE
  return(RMSE(predicted_ratings, validation_time$rating))
}

# Find out which is the best lambda value that minimises RMSE
lambdas <- seq(0, 10, 0.5)
rmses_y <- sapply(lambdas, movie_user_genre_year_reg_rmse)

# Plot RMSE for different lambas
qplot(lambdas, rmses_y, xlab = 'lambda', ylab = 'rmse')
```

```
# Find out minimum lambda value
lambda_min <- lambdas[which.min(rmses_y)]
```

Since the minimum RMSE is obtained with a $\lambda = 5.5$. Let's run the regularised model with that value as previously did with other regularisation effects. The RMSE result obtained can be observed in the Table 11.

```
# Re-calculate the model with the lambda that gives  the minimum RMSE
model_movie_user_gen_yr_reg_bias_rmse <- movie_user_genre_year_reg_rmse(lambda_min)
meet_goal <- ifelse(model_movie_user_gen_yr_reg_bias_rmse < min_rmse,
                    "Yes", "No")

# Collect the results in the tibble
rmse_results <- bind_rows(rmse_results,
                          tibble(Method = "Movie + User +
                                          Genre + Year Regularisation Effect",
```

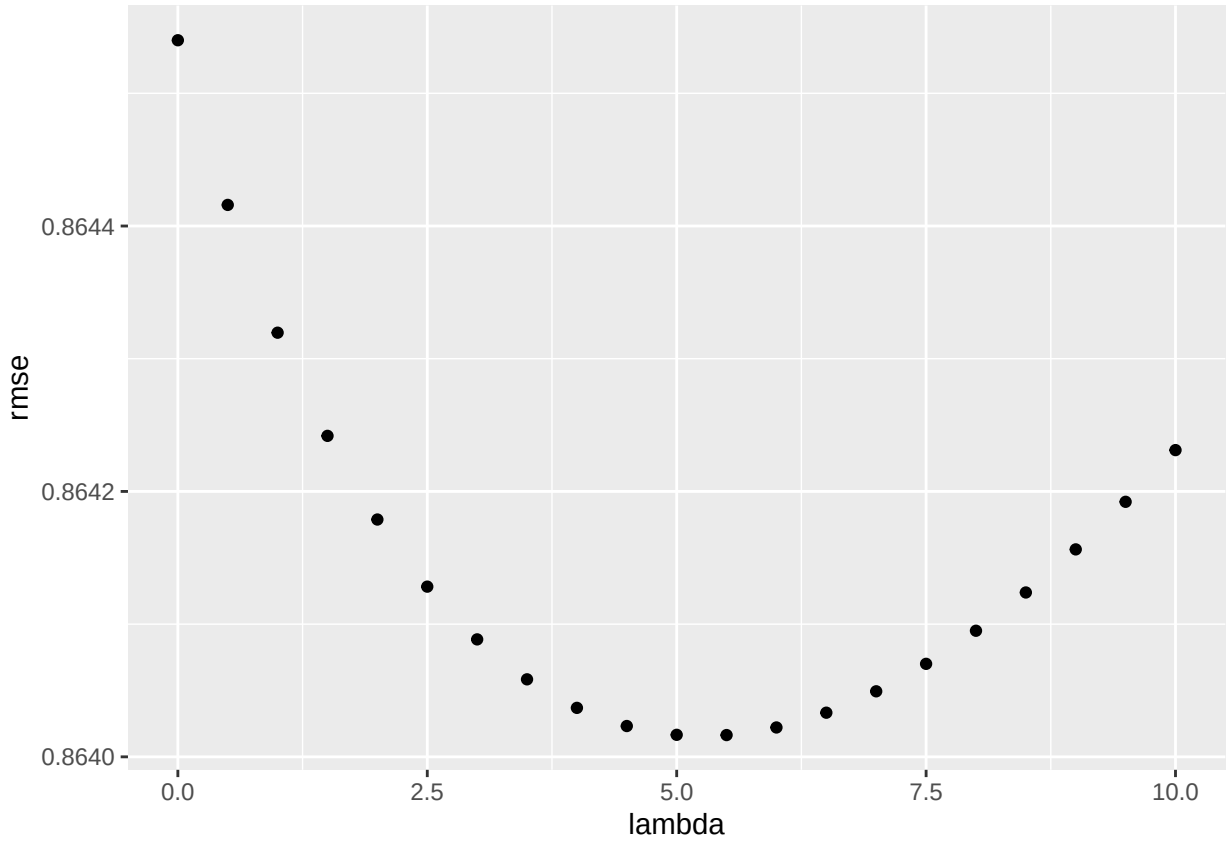

Figure 20: Regularisation plot for the movies with the user and movie effect accounting for the number of movies rated

```
                                    RMSE = model_movie_user_gen_yr_reg_bias_rmse,
                                    Meets.Goal = meet_goal))

# Print the table
knitr::kable(rmse_results,
             caption = "RMSE results for using
                        the average, the movie,
                        the user bias,
                        the user + movie bias and
                        the user + movie + genre bias",
             digits = 5, align = c('l','c','c')) %>%
             kableExtra::kable_styling(latex_options = "hold_position")
```

Table 11: RMSE results for using the average, the movie, the user bias, the user + movie bias and the user + movie + genre bias

Method	RMSE	Meets.Goal
Just the average	1.06120	No
Movie Effect	0.94391	No
User Effect	0.97834	No
Movie + User Effect	0.86535	No
Movie + User + Genre Effect	0.86495	No
Movie + User + Genre + Year Released Effect	0.86476	Yes
Movie + User + Genre + Year Rating Effect	0.86454	Yes
Movie + User Regularisation Effect	0.86482	Yes
Movie + User + Genre Regularisation Effect	0.86445	Yes
Movie + User + Genre + Year Regularisation Effect	0.86402	Yes

As we see, considering the movie, user, genre, time bias with the regularisation effect provides the best combinations of results.

1.5 Conclusions

We have developed a recommendation system that achieves the target set by building up on previous courses, as required from the assignment.

Our best approach consists in a regularisation effect that accounts for the biases on the movies, the users, the genre(s) selected and also the difference between years in which the movie was rated. We have also included other approaches that fall below the threshold.

The algorithm used is fairly simple but due to the lack of time and the scope of the work, it was decided not to pursue more complex approaches.

1.5.1 Limitations

We only evaluated one type of model in this work. Different algorithms could be tried. Due to the size of the data, some problems were encountered when working with it. Using a random trees or random forest approach could be problematic in that regard.

Another limitation we had was that we did not segment the genres appropriately. Maybe a better estimation could be obtained if dummy columns were created to captured if the model belongs to certain genre or not. In that way, probably a k-nearest neighbors approach could be applied if the other variables are also scaled.

1.5.2 Future works

Future works could be centered on using a similar approach than the *recommenderlab* package uses, by making use of sparse matrices and an attempt to use classification by items and also by users. Other users have successfully used the **reco** package to obtain much lower RMSEs than the ones provided in this work.