Capstone-MovieLens

Manuela Werkle

28th June 2021

1. Introduction

1.1 Description of the dataset

The MovieLens dataset is a dataset of movie recommendations done between January 1995 and January 2009. It was released in 1997 by GroupLens Research of the University of Minnesota. Users can score movies with a 0 until 5 star rating on a website (https://movielens.org/). You can rate movies, which you already know and based on your entries, MovieLens will suggest you further movies. For every user, MovieLens predicts the ratings of a user for any given movie. Based on these predictions, the system recommends movies to the user, which are assumed to be rated highly by the user.

The dataset which is used in this project is the 10M dataset from the grouplens website (https://grouplens.org/datasets/movielens/10m/) It contains 9 million ratings done for 10'000 movies by 72'000 users.

The dataset in this project contains of the following columns:

- userID: unique identification number of a user
- movieID: unique identification number of a movie
- rating: user rating of the corresponding movie
- timestamp: timestamp, when the user rating was done
- title: title of the movie (including it's year of release)
- genre: different genres or categories, where the movie can be associated. All assigned genres are separated with a pipe. Based on the file description on the website, there are the following genres available for all movies:
  - Action
  - Adventure
  - Animation
  - Children's
  - Comedy
  - Crime
  - Documentary
  - Drama
  - Fantasy
  - Film-Noir
  - Horror
  - IMAX
  - Musical
  - Mystery
  - Romance
  - Sci-Fi
  - Thriller
  - War
  - Western

1.2 Summary in regards to the goal of the project and key steps that were performed

In this project we will develop an own algorithm to predict movie ratings. To start our work, we pull the 10M MovieLens dataset directly from the MovieLens website and save it into a temp file. Afterwards we split the dataset into two datasets. The algorithm will be developed using the provided edx set (this is our training set). After the development, the final algorithm will then be used to predict already given movie ratings in the provided validation set (this is our test set).

RMSE will be used to check how close these predictions are compared to the real predictions in the validation set.

2. Methods and Analysis

2.1 Process and techniques used

We first need to clean our data. This will give us a better possibility of creating several analysis based on the different characteristics of the data. With the help of the data exploration we will then identify key insights to find effects, which will be used in our model creation using RSME.

2.2 Data Cleaning

If we look at the edx dataset, we recognize, that we first need to clean some data. This is necessary to make it easier to visualize and analyze the data.

2.2.1 Change format of the timestamp into a new column rating_date The value in the column 'timestamp' is the date, when the rating of the movie was done. It's formatted as the amount of seconds since 1st January 1970. We will convert it into the data type datetime and add a new column called 'rating_date' for this value:

```
# add new column for the rating date for both train and test set
edx <- edx %>%
  mutate(rating_date=as_datetime(timestamp))

validation <- validation %>%
  mutate(rating_date=as_datetime(timestamp))
```

2.2.2 Extract the year from the movie title The release year of the movie is included in in the movie title. It needs to be extracted and converted into a numeric data type. The new column for this value will be called 'movie_year':

```
# add new column for the movie release year for both train and test set
# The year is extracted as value between the brackets
edx <- edx %>%
  mutate(movie_year=as.numeric(str_replace_all(str_replace_all(str_extract(title,
  "[/(]\\d{4}[/)]$"),"\\(",""),"\\)","")))

validation <- validation %>%
  mutate(movie_year=as.numeric(str_replace_all(str_replace_all(str_extract(title,
```

2.2.3 Movie Age We will add a new column 'movie_age' to show how old the movie was, when the rating was done.

```
# add new column for the movie age for both train and test set
# movie age is the difference between movie release year in column movie_year
# and the year when the rating was done
edx <- edx %>%
  mutate(movie_age=year(rating_date)-movie_year)

validation <- validation %>%
  mutate(movie_age=year(rating_date)-movie_year)
```

2.2.4 Amount of genres to which the movie was assigned Movies are assigned to one or more genre. We will add a column called 'amount_genre' to specify to how many different genres a movie was assigned to. The less different genres are mentioned, the more specific is a movie.

```
# add new column for the amount of genre, different genres are separated with a pipe
# we will count the pipes and add 1 to have the correct value
edx <- edx %>%
  mutate(amount_genres=as.numeric(str_count(genres,fixed("|")))+1)

validation <- validation %>%
  mutate(amount_genres=as.numeric(str_count(genres,fixed("|")))+1)

mean(edx$amount_genres)
```

```
## [1] 2.596809
```

On average, movies are assigned to 2.59 genres.

2.2.5 Check after data cleaning To check, if there are any NA's in our table, we will use this code:

```
anyNA(edx) # Check for NA's in our table
```

```
## [1] FALSE
```

The result is FALSE, which means, that all values are complete.

2.3 Data exploration and data visualization

To have a first overview of the dataset we will explore the data and create some visualizations to gain some insights of the edx data set.

2.3.1 Some general data analysis To have a first overview of the data we will create a general overview and then go deeper into the different topics later.

```
nrow(edx) # count rows in edx data set
```

```
## [1] 9000055
```

The dataset contains 9000055 rows. Each row stands for a movie rating of a user.

```
edx %>% summarize(n_users = n_distinct(userId), n_movies = n_distinct(movieId))
```

```
##   n_users n_movies
## 1   69878    10677
```

There are 69878 different users who rated 10677 different movies.

Here we can see further summarizing information about the data set:

```
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           rating_date
##  Length:9000055     Length:9000055     Min.   :1995-01-09 11:46:49
##  Class :character   Class :character   1st Qu.:2000-01-01 23:11:23
##  Mode  :character   Mode  :character   Median :2002-10-24 21:11:58
##                                        Mean   :2002-09-21 13:45:07
##                                        3rd Qu.:2005-09-15 02:21:21
##                                        Max.   :2009-01-05 05:02:16
##    movie_year     movie_age     amount_genres
##  Min.   :1915   Min.   :-2.00   Min.   :1.000
##  1st Qu.:1987   1st Qu.: 2.00   1st Qu.:2.000
##  Median :1994   Median : 7.00   Median :3.000
##  Mean   :1990   Mean   :11.98   Mean   :2.597
##  3rd Qu.:1998   3rd Qu.:16.00   3rd Qu.:3.000
##  Max.   :2008   Max.   :93.00   Max.   :8.000
```

```
average_rating <- mean(edx$rating) # calculation of average rating
median_rating <- median(edx$rating) # calculation of median rating
```

The average rating of all movies is 3.512 and the median is 4.

```
# Put all genres mentioned on the website into a vector
genres <- c("Action","Adventure","Animation","Children's","Comedy","Crime",
            "Documentary","Drama","Fantasy","Film-Noir","Horror","IMAX","Musical",
            "Mystery","Romance","Sci-Fi","Thriller","War","Western")
```

```
# Check how many genres there are
length(genres)
```

```
## [1] 19
```

The movies are assigned to different genres. There are overall 19 different genres available. Some movies are not assigned. A movie can be assigned to one or more genres at the same time.

2.3.2 Ratings

The average of all ratings is 3.512 and the median is 4. We'll create an overview to see how often a specific rating was done:

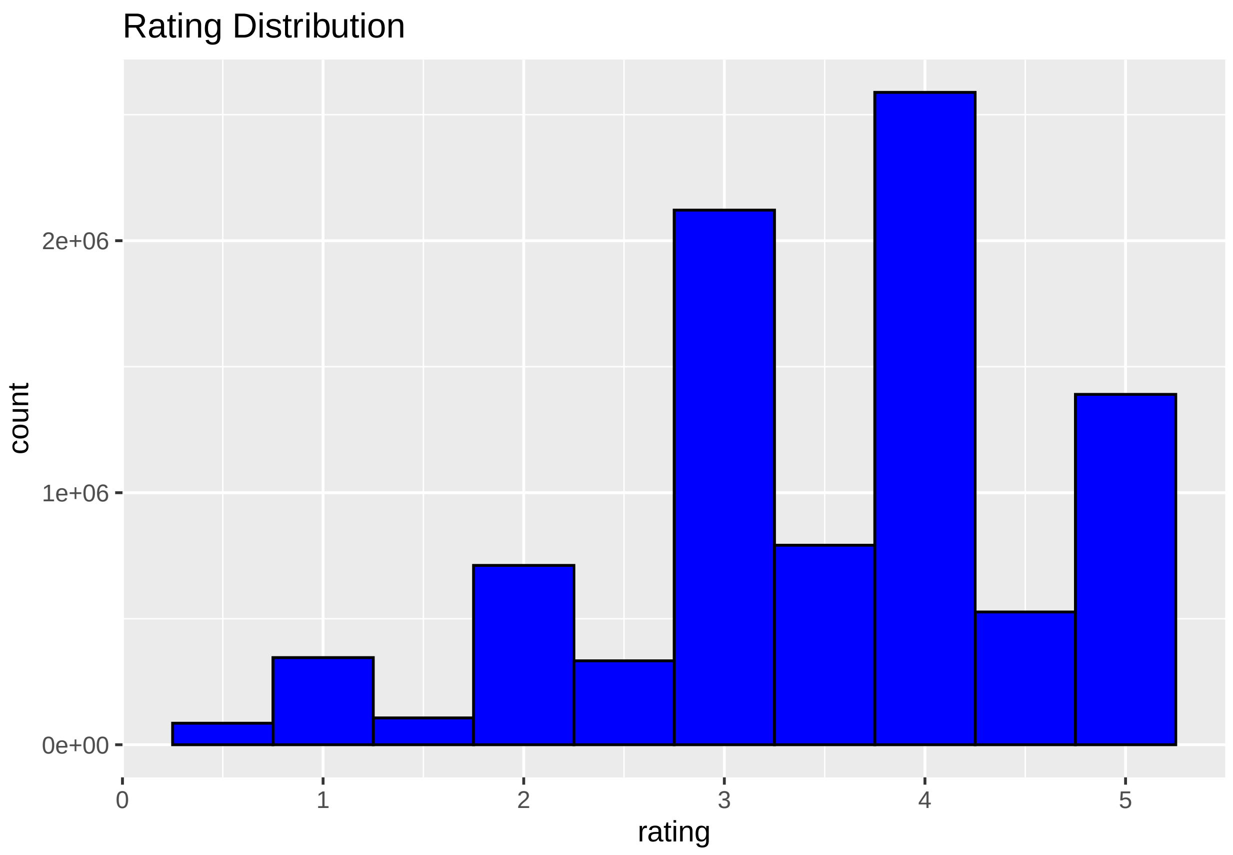
```
# take rating data and count how often one rating was done
analysis_rating <- edx %>%
  group_by(rating) %>%
  summarize(count_ratings = n()) %>%
  arrange(desc(rating)) # sort in descending order

analysis_rating # show rating distribition
```

```
## # A tibble: 10 x 2
##    rating count_ratings
##     <dbl>         <int>
##  1    5         1390114
##  2    4.5        526736
##  3    4         2588430
##  4    3.5        791624
##  5    3         2121240
##  6    2.5        333010
##  7    2          711422
##  8    1.5        106426
##  9    1          345679
## 10    0.5         85374
```

We see, that the rating, which occurs most, is the median value 4.

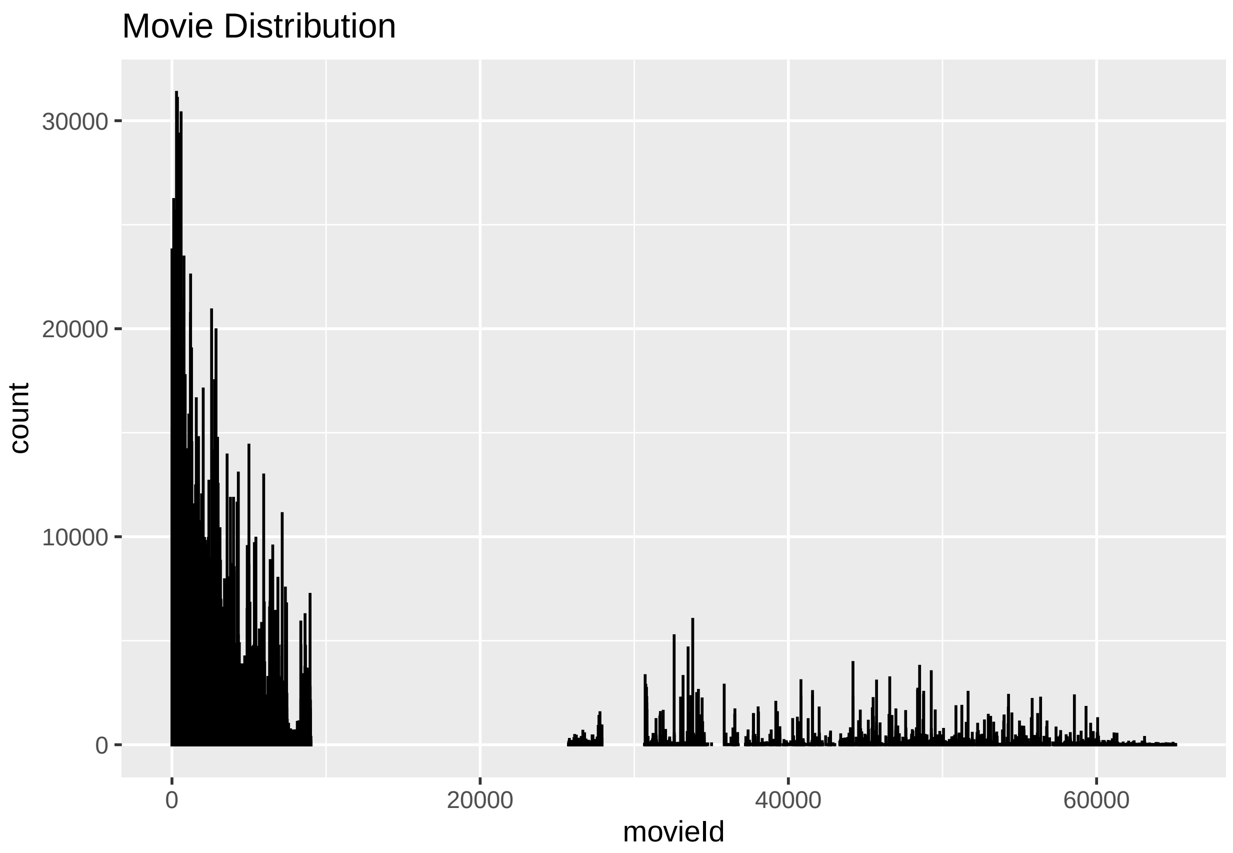
```
# plot the result into a histogram
 edx %>%
    ggplot(aes(rating)) +
    geom_histogram(binwidth=0.5, color="black", fill="blue") +
    ggtitle("Rating Distribution")
```



If we compare the distribution of the ratings, we see that there are more "good" ratings above 3 than below 3. This is not surprisingly, because normally users look movies, which are recommended to them or where they think that they like them.

2.3.3 Movies Now we will have a look at the different movies. We'll check, if every movie has been rated as many times as another movie:

```r
# plot in histogram how often a MovieId occurs in the data set,
# this is the amount of available ratings for this movie.
  edx %>%
    ggplot(aes(movieId)) +
    geom_histogram(binwidth=0.5, color="black", fill="blue") +
    ggtitle("Movie Distribution")
```



Based on this plot we see, that some movies have more ratings than other movies. There are some movies with really many ratings. It seems that these are the blockbusters, movies which are quite famous and so watched by many users.

```
# count how often a movie was rated, including the average rating
analysis_movies <- edx %>%
  group_by(title) %>%
  summarize(num_ratings = n(), average_rating = mean(rating)) %>%
  arrange(desc(average_rating))

# Filter for movies, which have only 1 rating done by 1 user
nrow(subset(analysis_movies, num_ratings == 1))
```

```
## [1] 126
```

There are only a small number of movies, which have only one rating. All other movies besides of these 126 have at least 2 ratings or more.

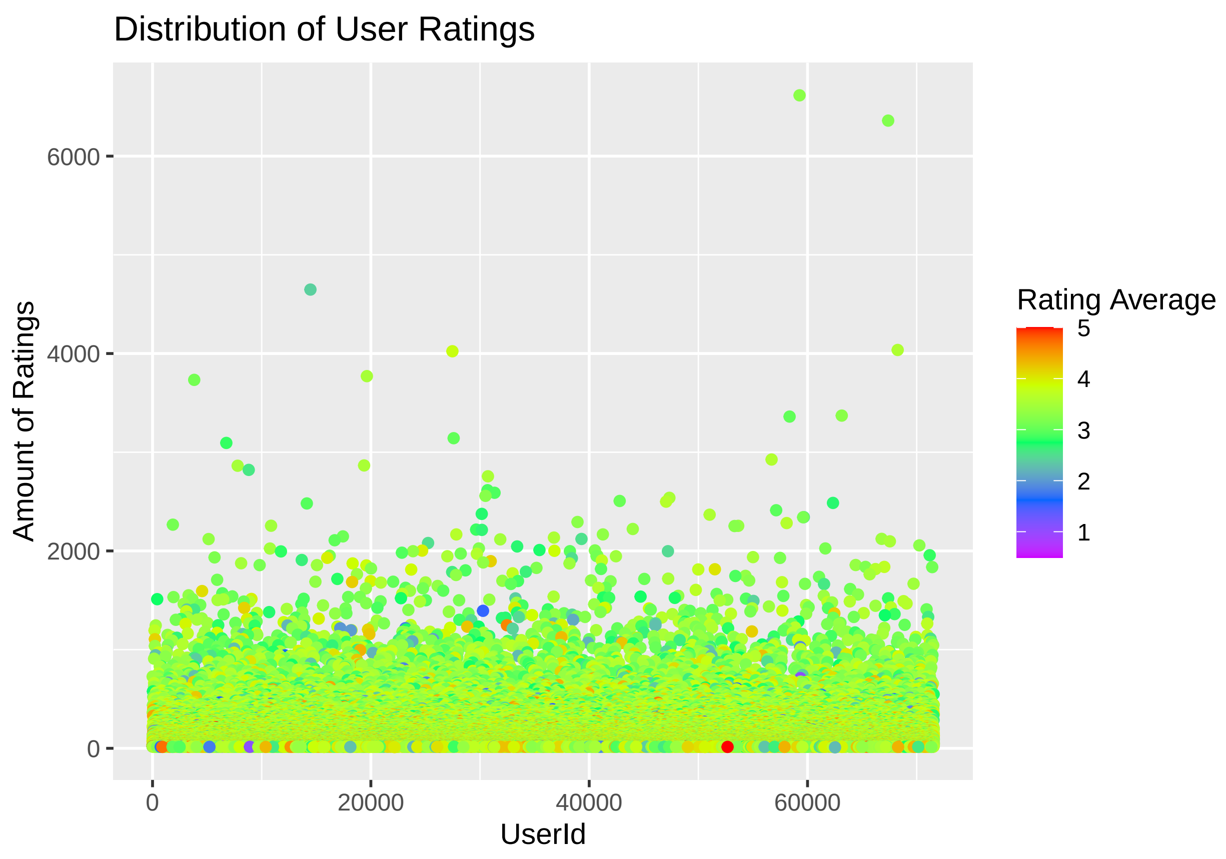
2.3.4 Users We will now check the amount of ratings and the average ratings of the different users. Overall we have 69878 users who did 9000055 ratings. This would mean, that a user would have done 128 ratings in average.

```
# Group by userId to show the amount of ratings done by this user
# including the average rating the user did for all his rated movies
users <- edx %>%
  group_by(userId) %>%
  summarize(num_ratings = n(), average_rating = mean(rating)) %>%
  arrange(desc(num_ratings))

head(users)
```

```
## # A tibble: 6 x 3
##   userId num_ratings average_rating
##    <int>       <int>          <dbl>
## 1  59269        6616           3.26
## 2  67385        6360           3.20
## 3  14463        4648           2.40
## 4  68259        4036           3.58
## 5  27468        4023           3.83
## 6  19635        3771           3.50
```

```
users %>%
  ggplot(aes(userId, num_ratings, color = average_rating)) +
  geom_point() +
  scale_color_gradientn(colours = c("#CC00FF", "#0066FF",
                                    "#00FF66", "#CCFF00", "#FF0000")) +
  labs(x = "UserId",
       y = "Amount of Ratings",
       title = "Distribution of User Ratings",
       color = "Rating Average")
```



Based on this plot, we see that most users have an own average rating which is green (between 3 and 4) which is near to the overall average rating. There's a only variation for users who have rated less movies. The highest amount of ratings done by a user is 6616 ratings. This is far away of the average 128, which means that there are other users who only rated a few movies. We can identify these users in the plot too.

2.3.5 Genre combinations We will now check for the average ratings of the different genres or genre combinations:

```
# group by genres, which can also be a combination of different genres,
# then count the amount and calculate the average rating
analysis_genres <- edx %>%
  group_by(genres) %>%
  summarize(num_ratings = n(), average_rating = mean(rating)) %>%
  arrange(desc(average_rating))

analysis_genres <- analysis_genres %>%
  mutate(amount_genres=as.numeric(str_count(genres,fixed("|")))+1)

head(analysis_genres) # display single genres
```

```
## # A tibble: 6 x 4
##   genres                          num_ratings average_rating amount_genres
##   <chr>                                 <int>          <dbl>         <dbl>
## 1 Animation|IMAX|Sci-Fi                     7           4.71             3
## 2 Drama|Film-Noir|Romance                2989           4.30             3
## 3 Action|Crime|Drama|IMAX                2353           4.30             4
## 4 Animation|Children|Comedy|Crime        7167           4.28             4
## 5 Film-Noir|Mystery                      5988           4.24             2
## 6 Crime|Film-Noir|Mystery                4029           4.22             3
```

We see, that the best rated genres are a combination of different genres. If we filter for single genres - means really specific movies which are only assigned to one genre at the same time - we see that Film-Noir and Documentary have got the best average ratings and the worst rated genres belong to Children and IMAX.

```
# Check for all movies, which have only been assigned to 1 specific genre
subset(analysis_genres, amount_genres == 1)
```

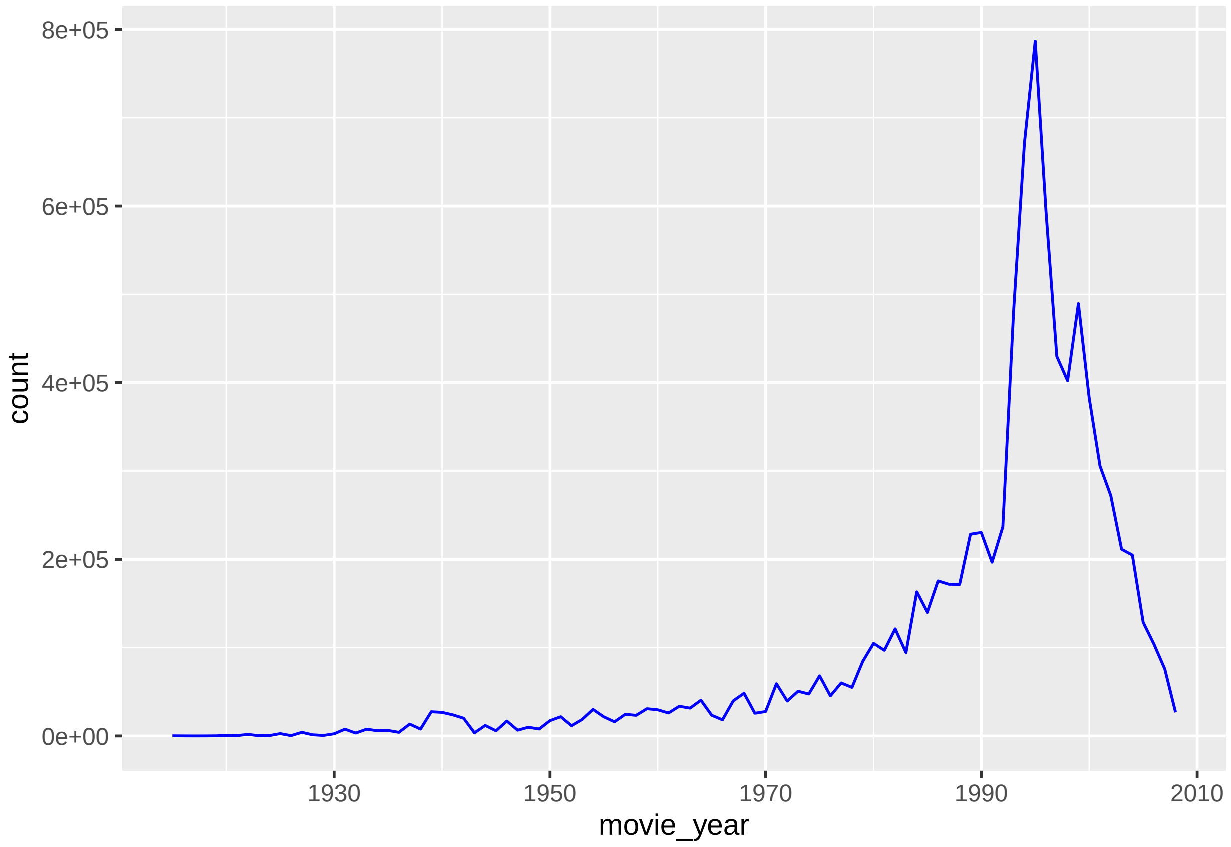
```
## # A tibble: 20 x 4
##    genres             num_ratings average_rating amount_genres
##    <chr>                    <int>          <dbl>         <dbl>
##  1 Film-Noir                 1575           3.83             1
##  2 Documentary              70041           3.82             1
##  3 Drama                   733296           3.71             1
##  4 War                       2300           3.67             1
##  5 (no genres listed)           7           3.64             1
##  6 Western                  15300           3.54             1
##  7 Thriller                 94662           3.53             1
##  8 Fantasy                     86           3.51             1
##  9 Musical                   3851           3.45             1
## 10 Romance                   8410           3.27             1
## 11 Comedy                  700889           3.24             1
## 12 Crime                     3197           3.23             1
## 13 Mystery                   246           3.06            1
## 14 Animation                 329           3.01            1
## 15 Adventure                2276           2.95            1
## 16 Action                  24482           2.94            1
## 17 Sci-Fi                  10125           2.93            1
## 18 Horror                  68738           2.88            1
## 19 Children                  745           2.49            1
## 20 IMAX                       14           2.32            1
```

But because there are too less movies only assigned to one single genre, we won't consider this insight in the next steps.

2.3.6 Movie Year We'll now check the distribution of the movie release years:

```
# Check how many movies have been release in the different year (movie year)
analysis_year <- edx %>%
  select(movieId, movie_year) %>%
  group_by(movie_year) %>%
  summarise(count = n())  %>%
  arrange(movie_year)

analysis_year %>%
  ggplot(aes(x = movie_year, y = count)) +
  geom_line(color="blue") +
  xlim(min(edx$movie_year),max(edx$movie_year))
```

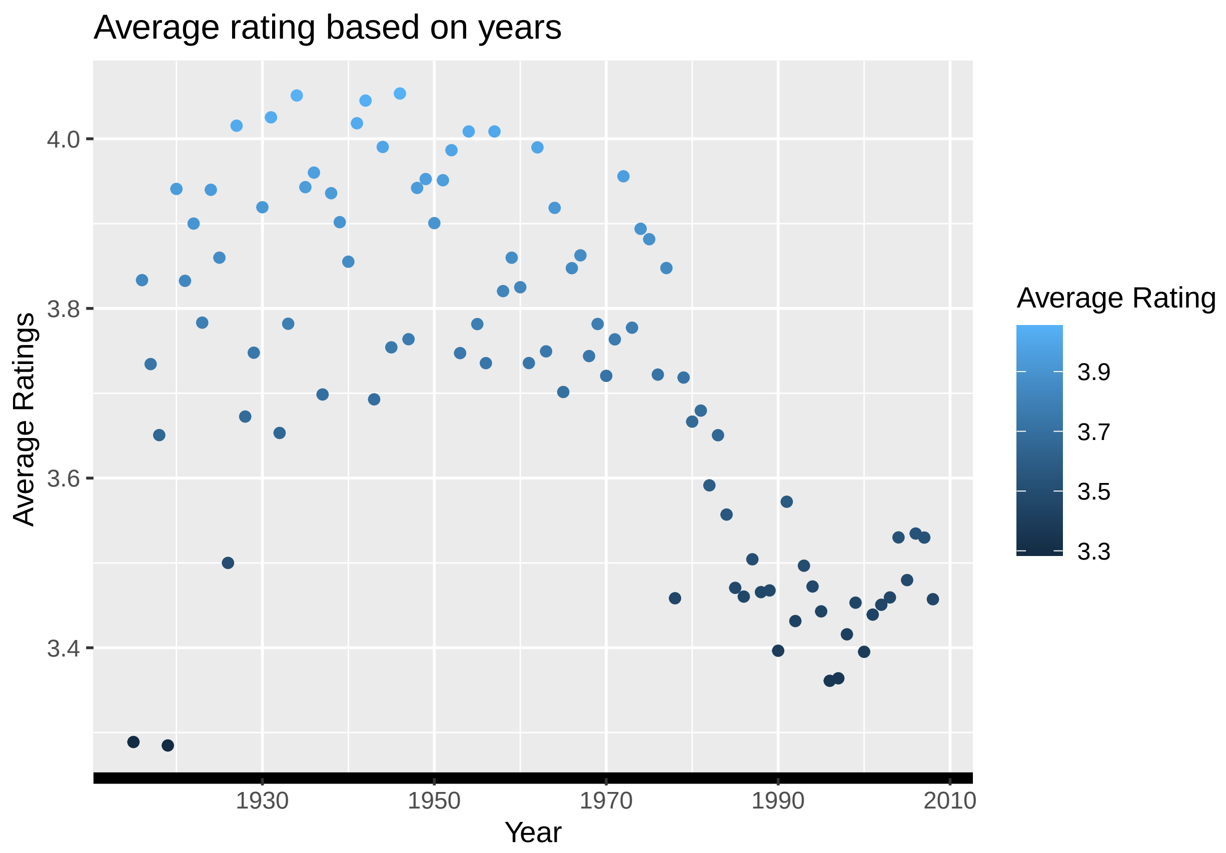


The oldest movies are from 1915 and the newest movies are from 2008.

To analyse this a little bit more deeper, we will include the average rating with different blue colors:

```
# Group by movie year again and include the average rating to see the
# average rating for every release year of the different movies
analysis_year2 <- edx %>%
  group_by(movie_year) %>%
  summarize(num_ratings = n(), average_rating = mean(rating)) %>%
  arrange(desc(num_ratings))

ggplot(analysis_year2) +
  geom_point(aes(x=movie_year, y=average_rating, colour=average_rating))+
  labs(title="Average rating based on years",
       x = "Year",
       y = "Average Ratings",
       colour = "Average Rating") +
  theme(axis.line.x = element_line(size=2))
```

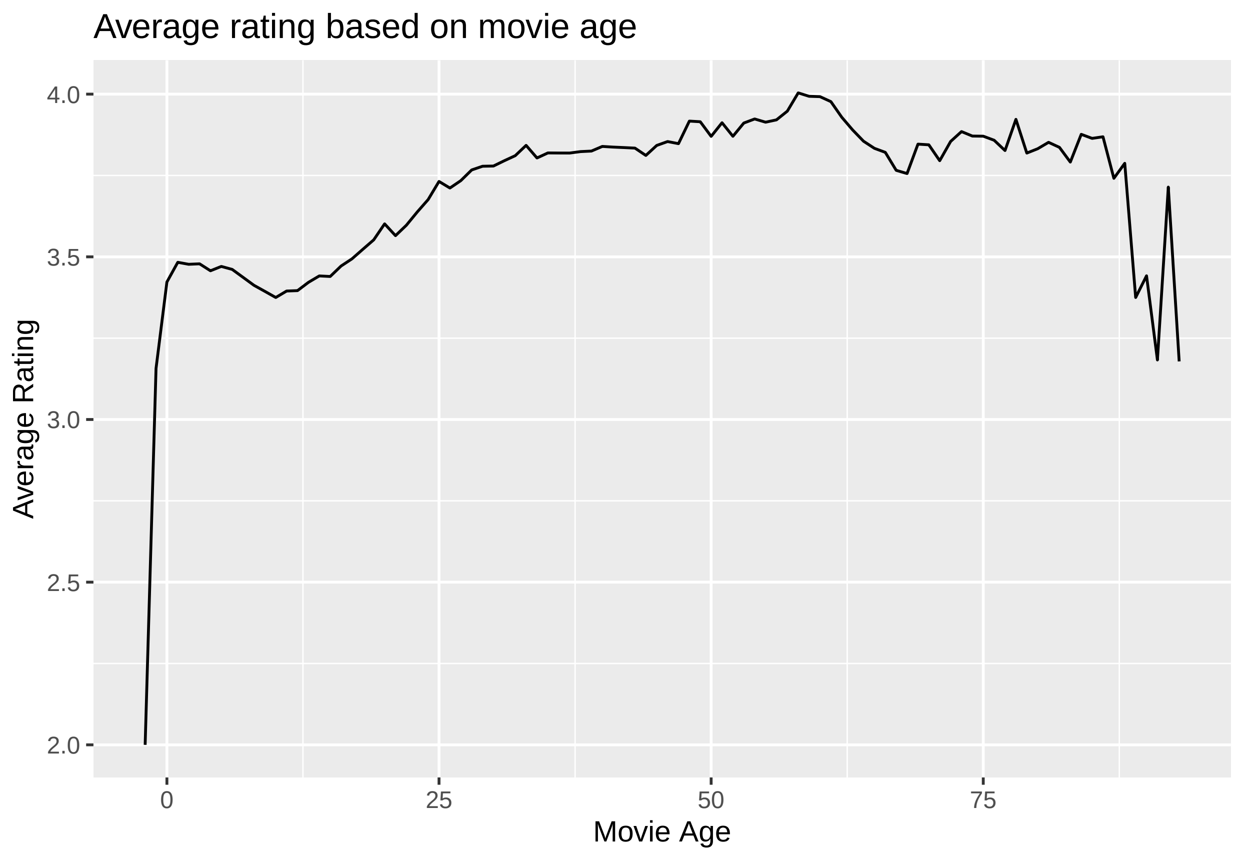


Based on this plot we can see, that the movies before around 1985 are rated better than movies, which are released afterwards.

2.3.7 Movie age We'll now check the amount and average ratings done based on the age of a movie at the time, when the rating was done:

```r
# Check the movie age to the time when the rating was done and show the amount
# of rating done for a specific age and the average rating
analysis_movie_age <- edx %>%
  group_by(movie_age) %>%
  summarize(num_ratings = n(), average_rating = mean(rating)) %>%
  arrange(desc(num_ratings))

analysis_movie_age %>%
  ggplot(aes(movie_age, average_rating)) +
  geom_line() +
  labs(x = "Movie Age", y = "Average Rating", title = "Average rating based on movie age")
```



Some movies are rated before their release date. This seems to be because of preview events. But in general we can see, that older movies have got a better rating. This only drops a little bit, when a movie is older than 90 years.

2.4 Insights gained

Based on the plots above we gain several insights:

- The amount of movies, which are released per year is increasing exponentially.
- Newer movies get more ratings.
- On average, movies are assigned to 2 or 3 genres (calculated value: 2.59).
- All genres show a similar rating behavior, but the best rated genre is Drama and Comedy.
- Especially after around 1985, there seem to be lower ratings.

- Users seem to have a positive bias, because most of their average ratings are above the mid-point of 2.5. But this might be to the fact, that users actually watch movies, which already have positive and good ratings.
- No users have less than 10 ratings. The highest amount of ratings has user with userID 59269 with 6616 ratings. (Which is quite impressive, because when checking the detailed data, this user watched 6616 movies in 8,5 years (time span between first and last rating of the user). This means in average, the user watched around 4 hours TV every day!!! ..)
- There are only 126 movies out of 10676, which are only rated once. All other movies are rated more often, which gives a better quality on the rating, because it presents the rating of more than one user.

Based on these insights we can identify differences in the ratings of a movie and also differences in the rating of the different users. Some users are more critical and avoid giving good ratings although for good movies, whereas other users often give full star ratings. In addition to that we also found out that the movie age and the movie release year seems to have an impact on the rating, which can be seen in above plots.

Based on these insights we can identify 4 effects, which will be considered in the following modeling approach: movie effect, user effect, movie age effect and movie release year effect.

2.5 Modeling approach

2.5.1 Simple Model with Average We will start with the prediction of the movie ratings regardless of all effects. For this, we will use the same rating for all movies, the average rating. A model which uses the same rating for all users and movies would look like this:

$$Y = \mu + \epsilon$$

In this formula, Y is the predicted rating, μ is the average rating and ϵ is the error.

```r
mu <- mean(edx$rating) # calculate mean and save in 'mu'
```

As already seen above, the average of all ratings is 3.512.

```r
# create model and compare predicted rating = mu with real ratings
naive_rmse <- RMSE(validation$rating,mu)

# Create tibble to show all results and add our first RMSE
rmse_results <- tibble(method = "Usage of average rating", RMSE = naive_rmse)
knitr::kable(rmse_results, caption = "Overview of the RMSE results")
```

Table 1: Overview of the RMSE results

method	RMSE
Usage of average rating	1.061202

The RMSE of 1.06 is far away of our goal to get a RMSE of < 0.86490. So we try to do better and include some effects to take more variability into account. Based on visualization above, we will include effects for movies, users, movie age and movie release year into the model.

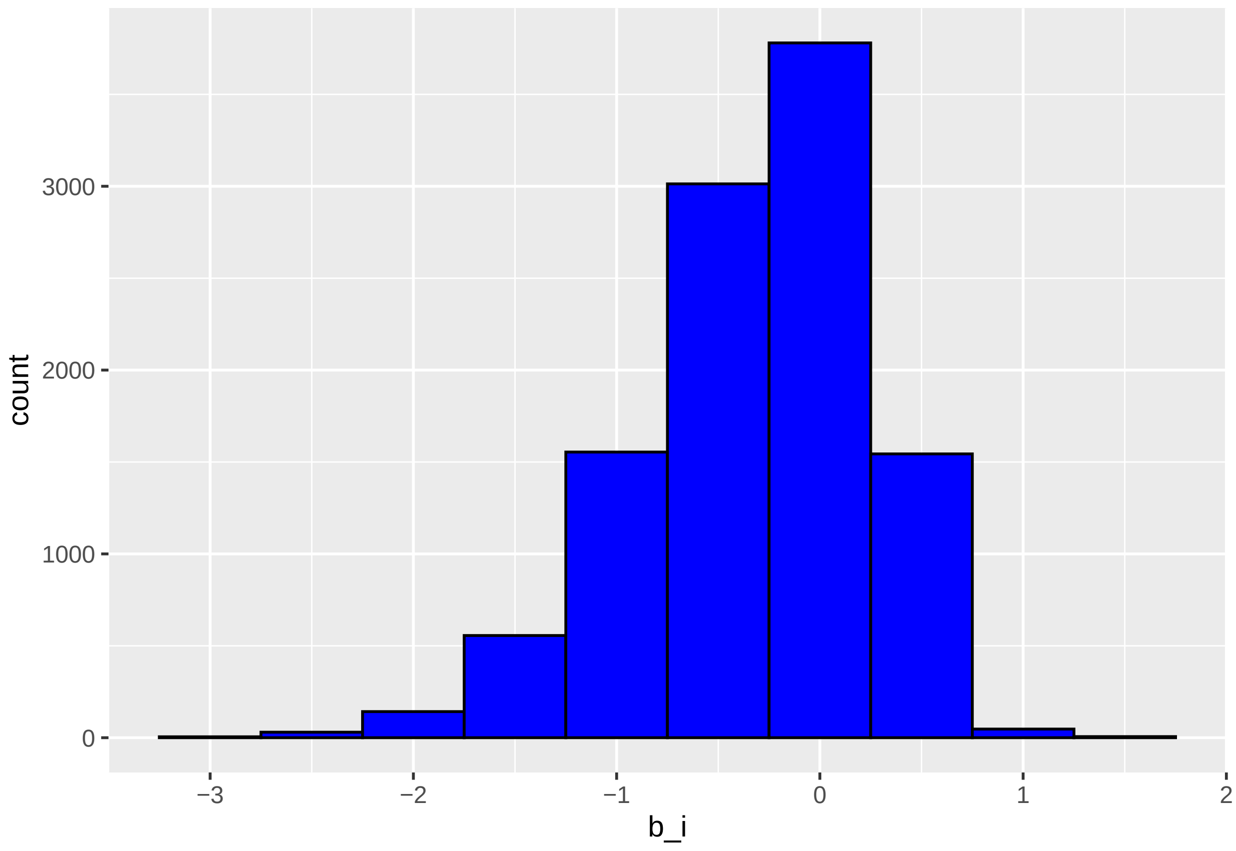
2.5.2 Include movie effect We know, that there are movies, which are rated better compared to other movies. That's why we will add a term b_i to the model, which indicates the rating for a movie i. This b_i is called movie effect.

$$Y = \mu + \epsilon + b_i$$

We know, that the least square estimate is just the average of $Y - \epsilon$ for each movie i, so we can model like this:

```r
# include movie effect into the model
movie_averages <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

movie_averages %>%
  ggplot(aes(b_i)) +
  geom_histogram(binwidth=0.5, color="black", fill="blue")
```



```r
# predict ratings based on the new model
predicted_ratings <- mu +
  validation %>%
  left_join(movie_averages, by='movieId') %>%
  .$b_i

# create model and compare predicted rating with real ratings
model_movie_effects <- RMSE(predicted_ratings, validation$rating)

# add movie effects result to the RMSE table
```

```
rmse_results <- bind_rows(rmse_results,
                          tibble(method = "Adding movie effects",
                          RMSE = model_movie_effects))
knitr::kable(rmse_results, caption = "Overview of the RMSE results")
```

Table 2: Overview of the RMSE results

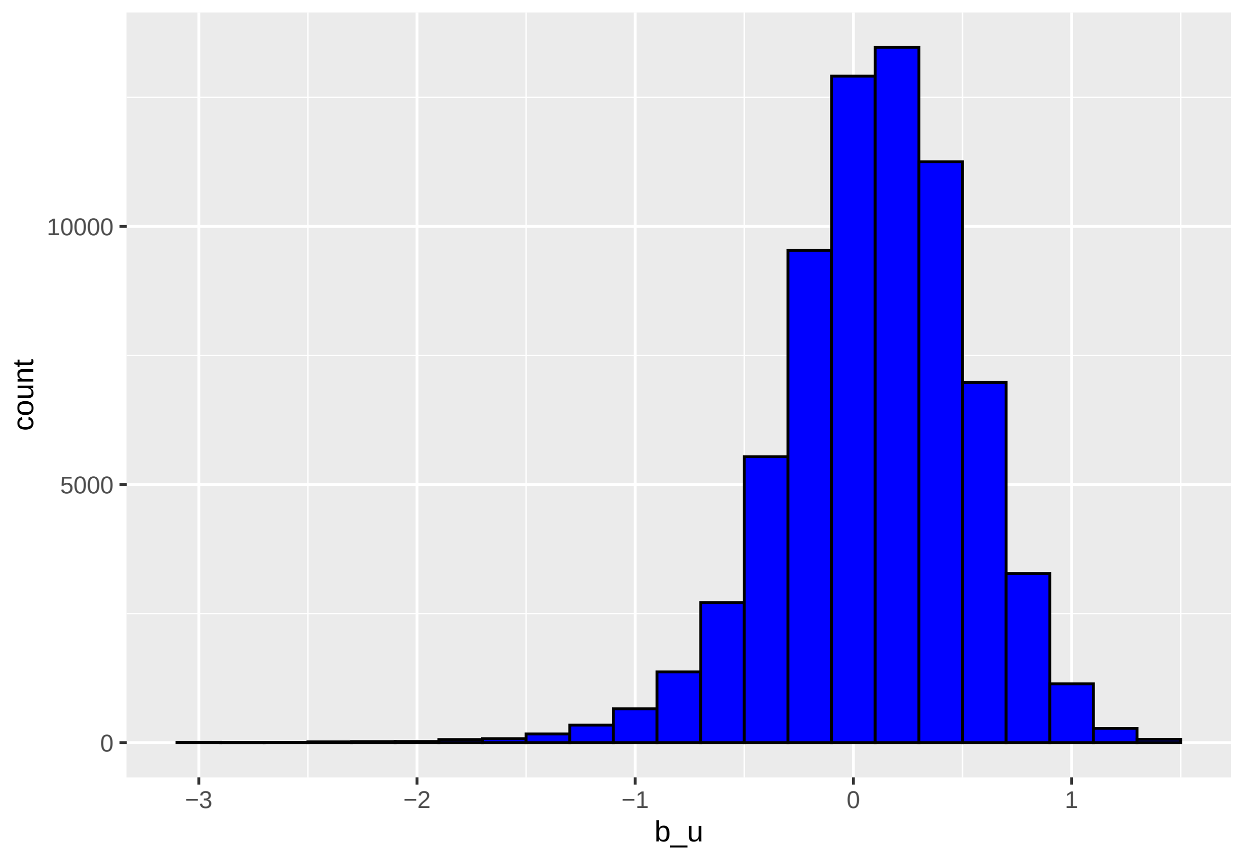
method	RMSE
Usage of average rating	1.0612018
Adding movie effects	0.9439087

Now we have an RMSE of 0.9439 which is better than before. But still far away from our goal < 0.86490. So we continue trying to do better with including more effects.

2.5.3 Include user effect We will now include the average rating for each user u. So our model will be extended to a new term / effect b_u indicating the rating for a specific user. This b_u is called user effect and our new formula will look like this:

$$Y = \mu + \epsilon + b_i + b_u$$

```
# include user effect into the model
edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(binwidth=0.2, color="black", fill="blue")
```



Including the user effect to the model will consider for example, that if critical users (these are the users with a negative b_u in the plot above) rate good movies, they will rather rate it with a 3 than with a 5.

```r
user_averages <- edx %>%
  left_join(movie_averages, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

# predict ratings based on the new model
predicted_ratings <- validation %>%
  left_join(movie_averages, by='movieId') %>%
  left_join(user_averages, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

# create model and compare predicted rating with real ratings
model_user_effects <- RMSE(predicted_ratings, validation$rating)

# add user effects result to the RMSE table
rmse_results <- bind_rows(rmse_results,
                          tibble(method = "Adding user effects",
                          RMSE = model_user_effects))
knitr::kable(rmse_results, caption = "Overview of the RMSE results")
```

Table 3: Overview of the RMSE results

method	RMSE
Usage of average rating	1.0612018
Adding movie effects	0.9439087
Adding user effects	0.8653488

Now we have an RMSE of 0.8653 which is better than before. But still a little bit away from our goal < 0.86490. So we continue trying to do better with including other effects.

2.5.4 Include movie age effect We will now include the average rating for the age of a movie a So our model will be extended to a new term / effect b_a indicating the rating at a specific movie age. This b_a is called movie age effect and our new formula will look like this:

$$Y = \mu + \epsilon + b_i + b_u + b_a$$

```r
# include movie age effect into the model
movie_age_averages <- edx %>%
  left_join(movie_averages, by='movieId') %>%
  left_join(user_averages, by='userId') %>%
  group_by(movie_age) %>%
  summarize(b_a = mean(rating - mu - b_i - b_u))

# predict ratings based on the new model
predicted_ratings <- validation %>%
  left_join(movie_averages, by='movieId') %>%
  left_join(user_averages, by='userId') %>%
  left_join(movie_age_averages, by='movie_age') %>%
  mutate(pred = mu + b_i + b_u + b_a) %>%
  .$pred

# create model and compare predicted rating with real ratings
model_movie_age_effects <- RMSE(predicted_ratings, validation$rating)

# add movie age effects result to the RMSE table
rmse_results <- bind_rows(rmse_results,
                          tibble(method = "Adding movie age effects",
                          RMSE = model_movie_age_effects))
knitr::kable(rmse_results, caption = "Overview of the RMSE results")
```

Table 4: Overview of the RMSE results

method	RMSE
Usage of average rating	1.0612018
Adding movie effects	0.9439087
Adding user effects	0.8653488
Adding movie age effects	0.8649038

Now we have an RMSE of 0.8649 which is only a little bit better than before. So we continue trying to do better with including another effect, the release year of the movie.

2.5.5 Include movie year (release year) effect We will now include the average rating for the release year of a movie y So our model will be extended to a new term / effect b_y indicating the rating for a specific movie release year. This b_y is called movie release year effect and our new formula will look like this:

$$Y = \mu + \epsilon + b_i + b_u + b_a + b_y$$

```r
# include movie release year effect into the model
movie_year_averages <- edx %>%
  left_join(movie_averages, by='movieId') %>%
  left_join(user_averages, by='userId') %>%
  left_join(movie_age_averages, by='movie_age') %>%
  group_by(movie_year) %>%
  summarize(b_y = mean(rating - mu - b_i - b_u - b_a))

# predict ratings based on the new model
predicted_ratings <- validation %>%
  left_join(movie_averages, by='movieId') %>%
  left_join(user_averages, by='userId') %>%
  left_join(movie_age_averages, by='movie_age') %>%
  left_join(movie_year_averages, by='movie_year') %>%
  mutate(pred = mu + b_i + b_u + b_a + b_y) %>%
  .$pred

# create model and compare predicted rating with real ratings
model_movie_year_effects <- RMSE(predicted_ratings, validation$rating)

# add movie release year effects result to the RMSE table
rmse_results <- bind_rows(rmse_results,
                          tibble(method = "Adding movie release year effects",
                          RMSE = model_movie_year_effects))
knitr::kable(rmse_results, caption = "Overview of the RMSE results")
```

Table 5: Overview of the RMSE results

method	RMSE
Usage of average rating	1.0612018
Adding movie effects	0.9439087
Adding user effects	0.8653488
Adding movie age effects	0.8649038
Adding movie release year effects	0.8646456

Now we have an RMSE of 0.8646 and with this result by adding the movie release year effect to our model we reached our goal of a RMSE <- 0.86490.

3. Results

3.1 Presentation of modeling results

Looking at the models above, we see that we received the best RMSE result by including the four effects for movies, users, movie age and movie release year to our average model. Our final result can be shown here:

```r
knitr::kable(rmse_results, caption = "Overview of the RMSE results")
```

Table 6: Overview of the RMSE results

method	RMSE
Usage of average rating	1.0612018
Adding movie effects	0.9439087
Adding user effects	0.8653488
Adding movie age effects	0.8649038
Adding movie release year effects	0.8646456

3.2 Discussion of the model performance

Checking the model performance we can see, that the calculations took longer when adding the significant improvements above. This is due to the fact, that there are more detailed information to consider into account using the left join statements for the calculation.

4. Conclusion

4.1 Brief summary of the report

Based on the data exploration we found several different insights based on which we defined the possible model effects. Hereby the most significant improvement was done by adding the movie and user effect whereas the movie age and movie release year effect only improved our result a little bit.

4.2 Limitations of the report

The data set is very long. For all calculations it's necessary to have enough RAM on the personal machine. From my personal point of view I faced a limitation here, because I'm only using quite an old tablet. So it happened several times that R Studio crashed due to too less memory. So some improvements were necessary so that the calculation finally worked. But to be more efficient it would be more comfortable to have a faster laptop or PC. Luckily we only used the 10M data set of Movielens here. I think it would not have been possible for me using a larger data set.

I also wanted to include the genre into the analysis using the separate_rows function to split all genre for one movie into single rows. But as this mostly resulted in the error message "Error: cannot allocate vector" because of too less memory, I removed it from my code again.

4.3 Future work

Although we reached our goal and received a final RMSE of < 0.86490 according to the project task, I think it should be possible to improve the RMSE further. First the effect of the genre should also be included. But as mentioned above, more memory would be necessary here. Further possible methods to optimize the model in future could be regularization and matrix factorization or the combination of these methods with the most significant model approaches above (movie and user effects).