MovieLens Project 2022

Anaiz

January 2022

#Introduction

The goal of this exercise is to predict the movie ratings in the validation set by using a training set and different models.

In order to reach the goal, machine learning algorithms are used.

1. Data preparation and analysis

At first, the code below is used for preparing the data base (test and validation set) and this code was provided by the edx team in the module capstone:

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.2     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
```

```
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

```r
library(tidyverse)
library(caret)
library(data.table)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")

# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

```r
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Before the real analysis starts, useful packages that could help with the analysis were installed (ex. ggplot2, dplyr,lubridate,validate,caret, etc..)

Once the packages were ready, a copy of edx was created:

```
edx1 <-edx
```

This step is not really needed, but I found it useful while doing the data analysis in case something did not work well.

1.1 Summary

Once ready, the first thing I did is look into the data set:

```
##      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
##
##
##
```

1.2 NA's

And controlled if there are any NA's in the database:

```
anyNA(edx1)
```

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

1.3 Conversion of timestamp

The timestamp class is "character" and in order to use the timestamp of when the rating was made, this parameter was transformed for the test and validation set as POSIXt:

```
edx1$timestamp <- as.numeric(edx1$timestamp)
edx1$timestamp <- as.POSIXct(edx1$timestamp, origin="1970-01-01")

validation$timestamp <- as.numeric(validation$timestamp)
validation$timestamp <- as.POSIXct(validation$timestamp, origin="1970-01-01")
```

Once done, the year of release was then moved into a new column and set as numeric:

```
edx1$ratingyear <-format(edx1$timestamp, format="%Y")

validation$ratingyear <-format(validation$timestamp, format="%Y")
```

1.4 Separate year of release into a new column

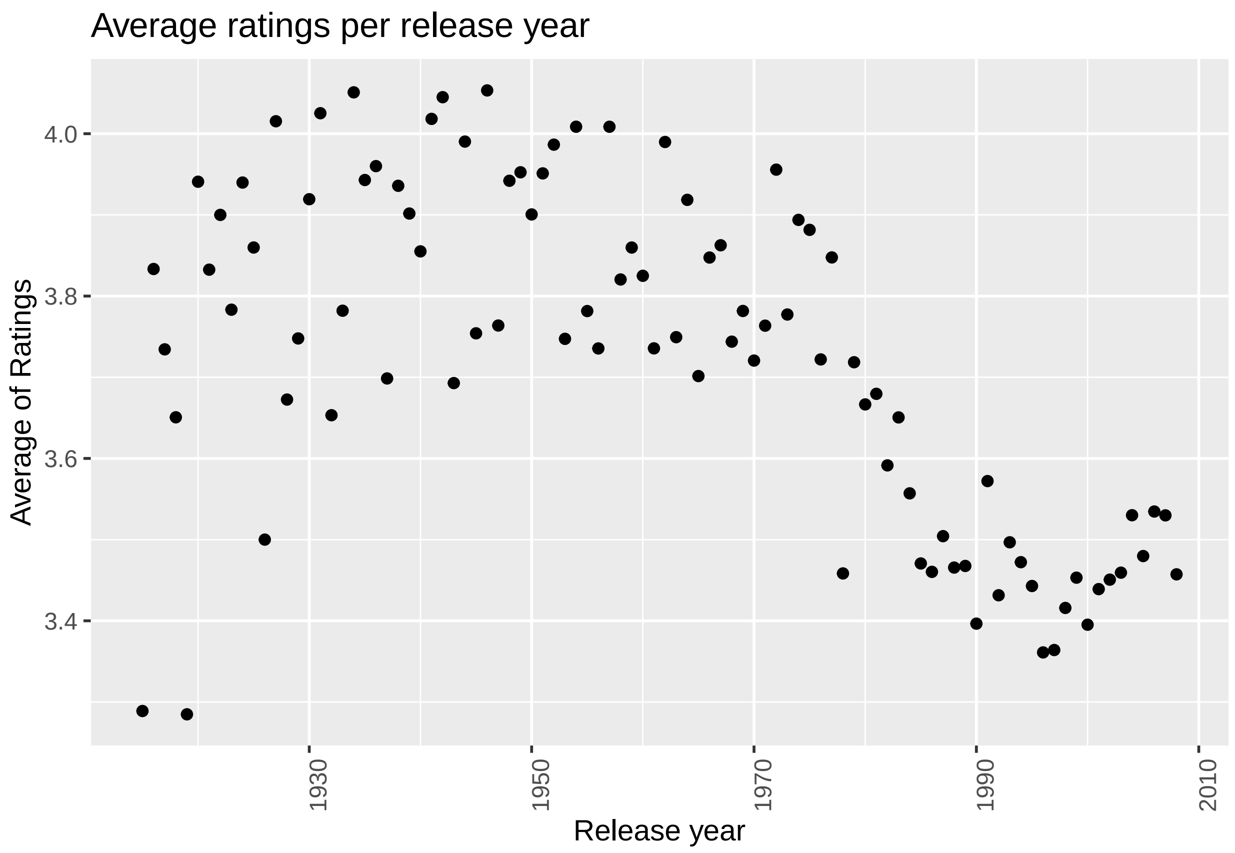
In order to see if the year of release has an impact on rating, I separated the year of release from the title of the movie into a new column and convert it in numeric:

```
edx1$releaseyear<-substr(edx1$title,nchar(as.character(edx1$title))-4,nchar(as.character(edx1$title))-1
edx1$title<-paste0(substr(edx1$title,1,nchar(as.character(edx1$title))-6))
```

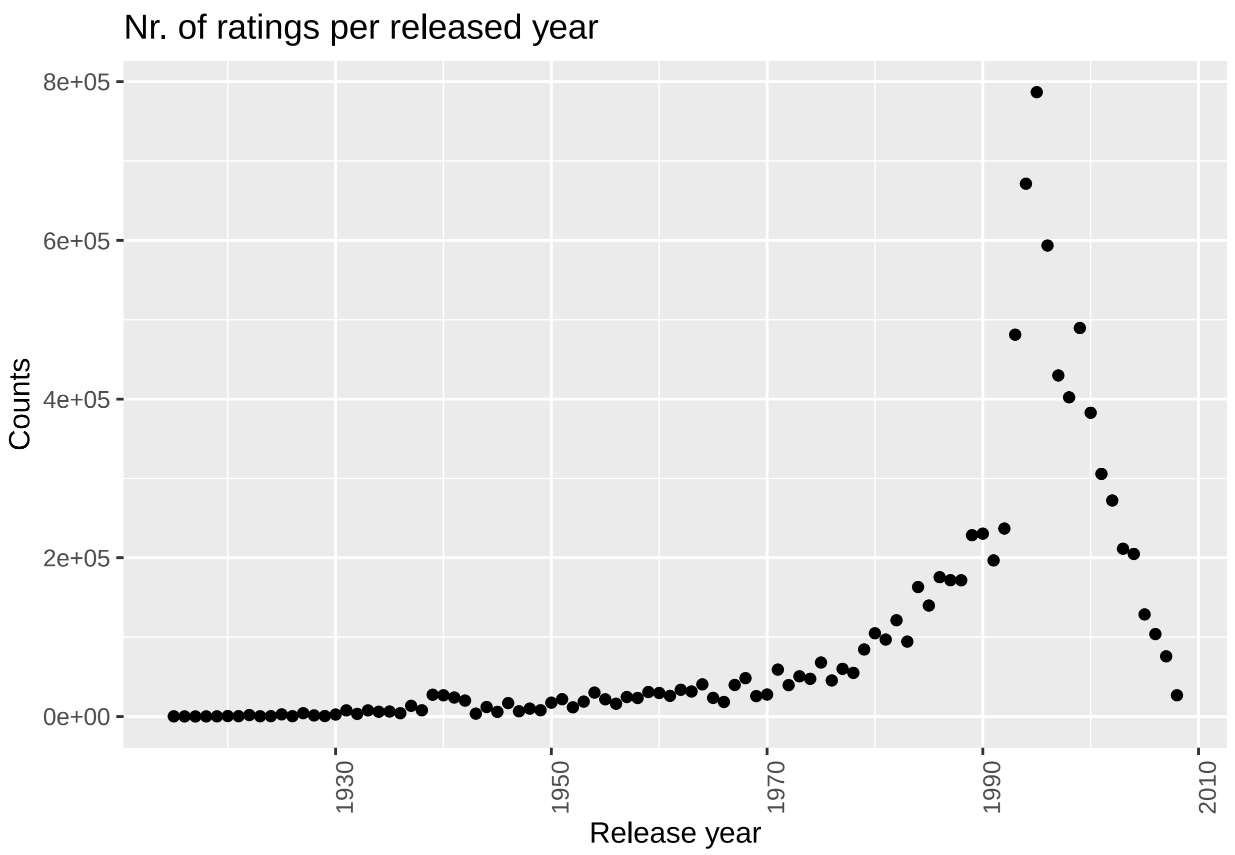
```
edx1$releaseyear<-as.numeric(edx1$releaseyear)

validation$releaseyear<-substr(validation$title,nchar(as.character(validation$title))-4,nchar(as.charac
validation$title<-paste0(substr(validation$title,1,nchar(as.character(validation$title))-6))

validation$releaseyear<-as.numeric(validation$releaseyear)
```

1.5 Observations for year of release

Following graphs were created in order to analyze the mean ratings on movies depending on the year of release and how many rating per year of release were made:



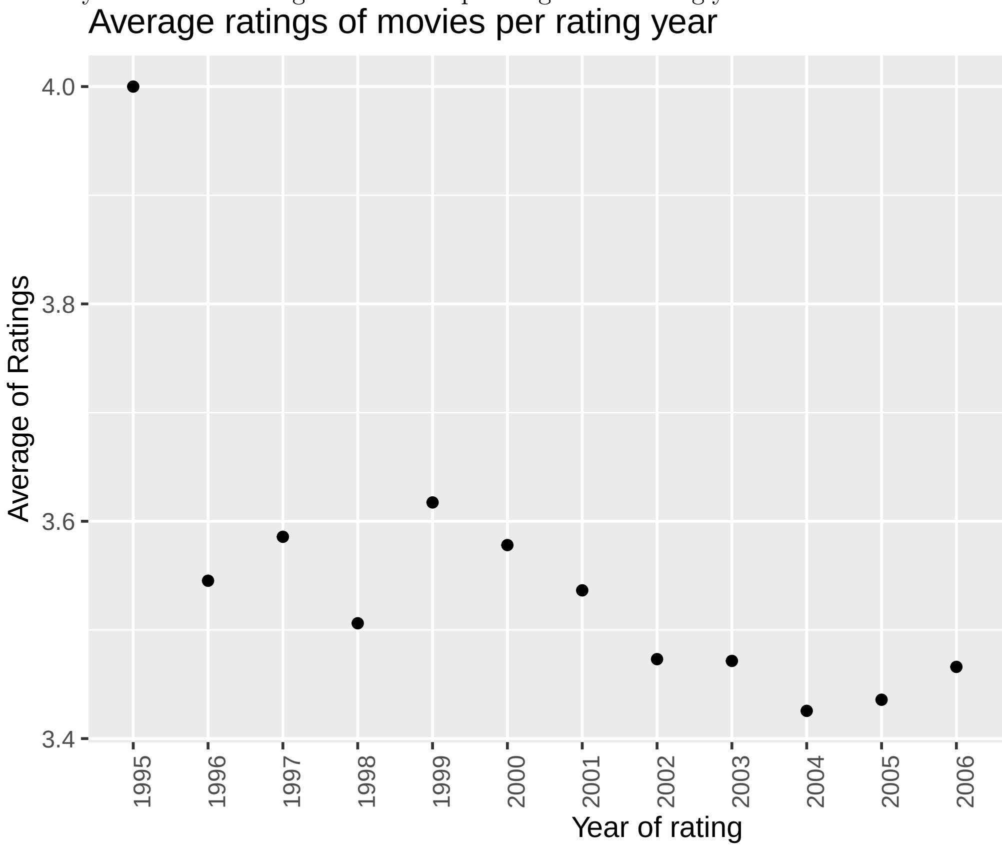
Movies released up to approx.1980 have an average rating of 3.8-3.9 while movies after 1980 have a lower rating.



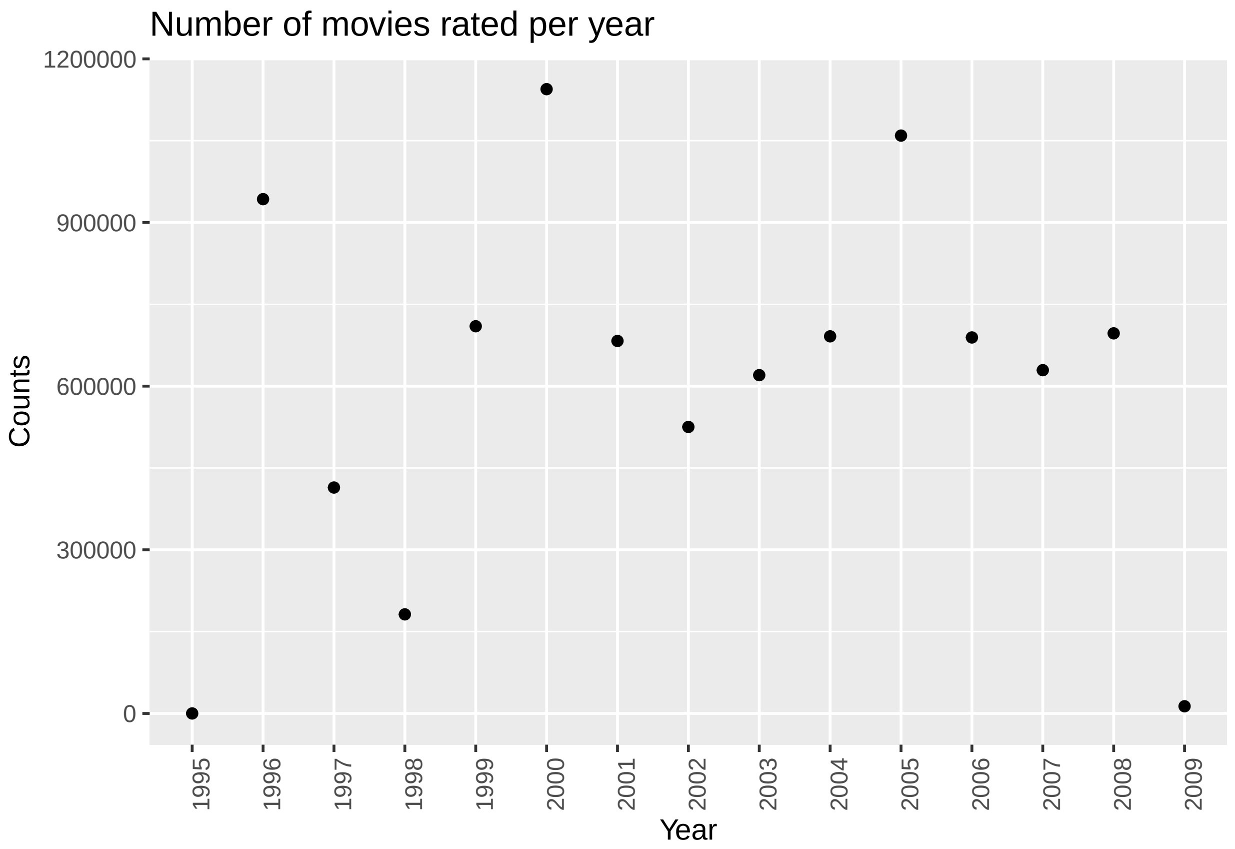
There are not a lot of ratings for movies before 1980. After 1980 the number of ratings per release year increased up to approx. 1995 and decreased once more.

1.6 Observations for year of rating

Following graphs were created in order to analyze the mean ratings of movies depending on the rating year and

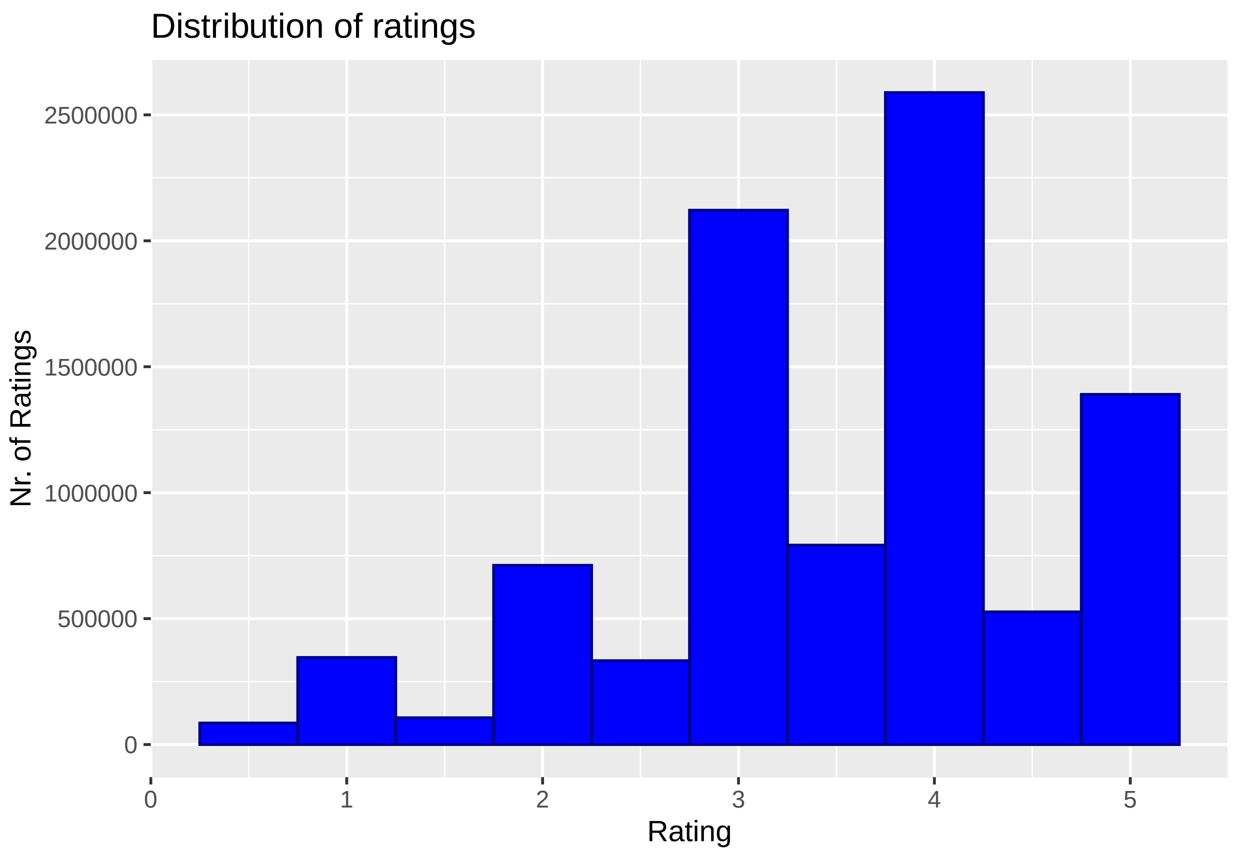


how many movies were rated per year:



With these graphs we can see that ratings started from 1995 and the average of rating in the first year is 4. From 1996 up to 2009 people rated in general the movies in average with 3.5. This lower average rating can be explained by the quantity of movies that were at disposal and the quantity of users that rated the movies.

1.7 Review rating distribution



It is interesting to see that people in general rated either with a "full" number (ex.3,4,5) and did not rate movies with ex. ".5" numbers.

2. Modelling analysis and Results

In this section, different models are displayed to calculate the RMSE. First you will find the basic one and then the complexity and parameters will increase. The variables used in the analysis are: Movie ID, User ID and Release year.

Please be aware, in the analysis "summarise" is written with "s" as with "z" the code did not work with my R and gave me an error each time I used it.

2.1 Residual Mean Squared Error function

The following function was used to determine the RMSE:

```r
RMSE <- function(true_prediction, predicted_prediction){
  sqrt(mean((true_prediction - predicted_prediction)^2))}
```

2.2 First model predicting rating regardless of other factors (user,year, etc..)

```r
mu_rating <- mean(edx1$rating)
mu_rating #3.512465
```

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

```r
first_rmse <- RMSE(validation$rating, mu_rating)
first_rmse
```

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

```r
#Create results dataframe with RMSE results
RMSE_Results <- data.frame(model="Baseline Model", RMSE=first_rmse)
```

The result for this baseline model is 1.061202.

2.3.1 Second model predicting movie effect

```r
movie_avg <- edx1 %>%
  group_by(movieId) %>%
  summarise(b_m = mean(rating - mu_rating))

predicted_ratings <- mu_rating + validation %>%
  left_join(movie_avg, by='movieId') %>%
  pull(b_m)

Movie_effect <- RMSE(predicted_ratings, validation$rating)

#Create results dataframe with RMSE results

RMSE_Results <- bind_rows(RMSE_Results,
                          data_frame(model="Movie effect Model",
                                     RMSE = Movie_effect))
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

In this model we try to predict the rating with the movie effect. The result for this movie effect model is 0.9439087.

2.3.2 Second model predicting the user effect

The same predictions as 2.3.1 were made but this time with the user.

```r
user_avg <- edx1 %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu_rating))

predicted_ratings <- mu_rating + validation %>%
  left_join(user_avg, by='userId') %>%
  pull(b_u)
User_effect <- RMSE(predicted_ratings, validation$rating)

#Create results dataframe with RMSE results

RMSE_Results <- bind_rows(RMSE_Results,
                          data_frame(model="User effect Model",
                                     RMSE = User_effect))
```

The result for this movie effect model is 0.978336.

2.3.3 Second model predicting the release year effect

The same predictions as 2.3.1 and 2.3.2 were made but with the release year.

```r
releaseyear_avg <- edx1 %>%
  group_by(releaseyear) %>%
  summarise(b_r = mean(rating - mu_rating))

predicted_ratings <- mu_rating + validation %>%
  left_join(releaseyear_avg, by='releaseyear') %>%
  pull(b_r)
Releaseyear_effect <- RMSE(predicted_ratings, validation$rating)

#Create results dataframe with RMSE results

RMSE_Results <- bind_rows(RMSE_Results,
                          data_frame(model="Release Year effect Model",
                                     RMSE = Releaseyear_effect))
```

The result for this movie effect model is 1.0500259.

Interestingly the movie effect has better RMSE, followed by user effect and release year.

In order to see if we can improve the RMSE, analysis for combined parameters was performed.

2.4.1 Third model predicting the movie effect and user to improve the RMSE

```r
user_avg <- edx1 %>%
  left_join(movie_avg, by='movieId') %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu_rating - b_m))

predicted_ratings <- validation %>%
  left_join(movie_avg, by='movieId') %>%
  left_join(user_avg, by='userId') %>%
  mutate(pred = mu_rating + b_m + b_u) %>%
  pull(pred)

Mo_U_effect <-RMSE(predicted_ratings, validation$rating)

#Create results dataframe with RMSE results

RMSE_Results <- bind_rows(RMSE_Results,
                          data_frame(model="Movie and User effect Model",
                                     RMSE = Mo_U_effect))
```

The RMSE result for movie and user effect decreased to 0.8653488.

2.4.2 Third model predicting the movie effect, user and release year to improve the RMSE

If we go a step further and add the release year to the model:

```r
user_avg <- edx1 %>%
  left_join(movie_avg, by='movieId') %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu_rating - b_m))

releaseyear_avg <- edx1 %>%
  left_join(movie_avg, by='movieId') %>%
  left_join(user_avg, by='userId') %>%
  group_by(releaseyear) %>%
  summarise(b_r = mean(rating - mu_rating - b_m - b_u))

predicted_ratings <- validation %>%
  left_join(movie_avg, by='movieId') %>%
  left_join(user_avg, by='userId') %>%
  left_join(releaseyear_avg, by='releaseyear') %>%
  mutate(pred = mu_rating + b_m + b_u + b_r) %>%
  pull(pred)

Combined_effect <-RMSE(predicted_ratings, validation$rating)

#Create results dataframe with RMSE results

RMSE_Results <- bind_rows(RMSE_Results,
                          data_frame(model="Combined effect Model",
                                     RMSE = Combined_effect))
```

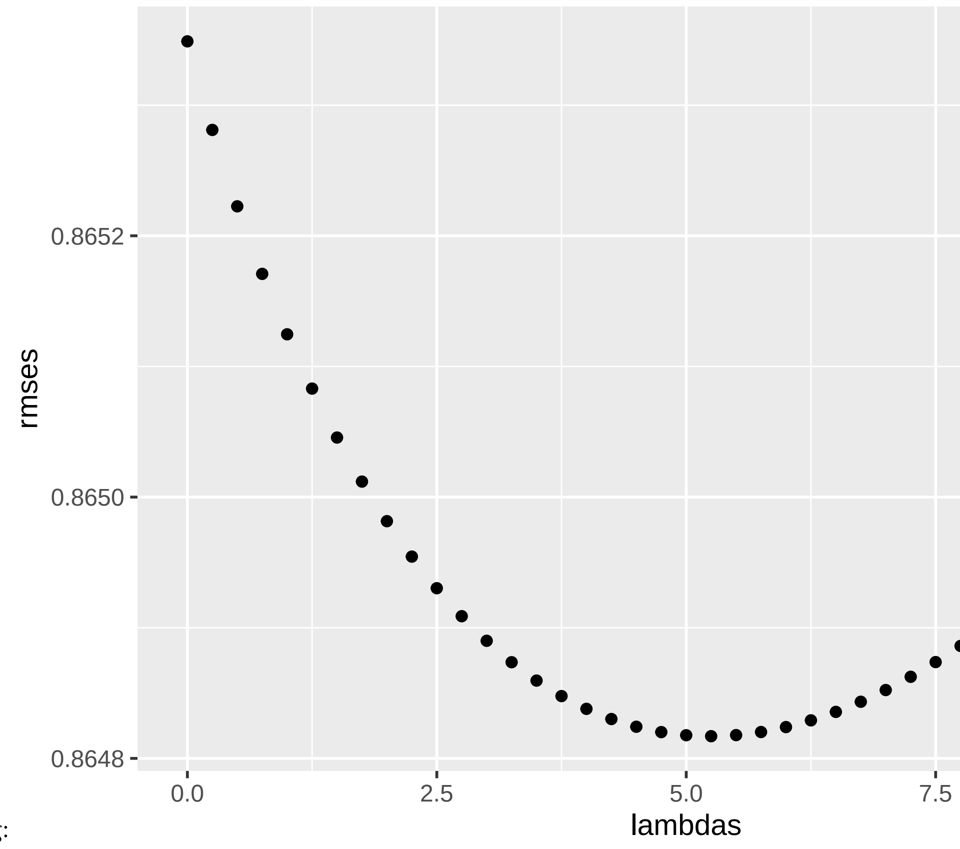
We see that the RMSE improved up to 0.8650043.

2.5 Regularized model, chose lambda (tuning parameter) for movie and user effect

In the data analysis, we have seen that movies released between 1915 and 1980 were not rated much. In order to obtain a more accurate RMSE and reduce the error possibilities, a regularization is needed. For this part, only movie and user effect were added.

Note: the below code takes some time

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

rmses <- sapply(lambdas, function(l){
  mu_rating <- mean(edx1$rating)

  b_m <- edx1 %>%
    group_by(movieId) %>%
    summarise(b_m = sum(rating - mu_rating)/(n()+l))

  b_u <- edx1 %>%
    left_join(b_m, by="movieId") %>%
    group_by(userId) %>%
    summarise(b_u = sum(rating - b_m - mu_rating)/(n()+l))

  predicted_ratings <- validation %>%
    left_join(b_m, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu_rating + b_m + b_u) %>%
    pull(pred)

  return(RMSE(predicted_ratings, validation$rating))
  })
```



The distribution of lambdas is the following:
By using following code, the optimal lambda was chosen

```
lambda1 <- lambdas[which.min(rmses)]
lambda1 #5.25
```

```
## [1] 5.25
```

Once the lambda was selected, it was added to the code in order to see if RMSE has improved.

```
movie_avg <- edx1 %>%
  group_by(movieId) %>%
  summarise(b_m = sum(rating - mu_rating)/(n()+lambda1), n_m = n())

user_avg <- edx1 %>%
  left_join(movie_avg, by='movieId') %>%
  group_by(userId) %>%
  summarise(b_u = sum(rating - mu_rating - b_m)/(n()+lambda1), n_u = n())

predicted_ratings <- validation %>%
    left_join(movie_avg, by='movieId') %>%
    left_join(user_avg, by='userId') %>%
    mutate(pred = mu_rating + b_m + b_u) %>%
    pull(pred)

reg_m_u_effect <- RMSE(predicted_ratings, validation$rating)

#Create results dataframe with RMSE results
```

```
RMSE_Results <- bind_rows(RMSE_Results,
                          data_frame(model="Reg. Model Movie and User effect",
                                     RMSE = reg_m_u_effect))
```

In this analysis we used only Movie and User effect combination and we saw that RMSE improved from 0.8653488 to 0.8648170 using the regularized model.

3. Conclusions

In general, we could see that using different models, the RMSE was improved:

model	RMSE
Baseline Model	1.0612018
Movie effect Model	0.9439087
User effect Model	0.9783360
Release Year effect Model	1.0500259
Movie and User effect Model	0.8653488
Combined effect Model	0.8650043
Reg. Model Movie and User effect	0.8648170

There is still room for more improvement if for ex. we use the regularized model with more effects like release year. This could be done for a future project.

4. References

https://files.grouplens.org/datasets/movielens/ml-10m-README.html

https://shiny.rstudio.com/articles/rmd-integration.html?_ga=2.69618128.1702490715.1641919411-132160490.1641919411

https://rafalab.github.io/dsbook/large-datasets.html