Data Science in R - MovieLens Rating Prediction

Wan Hervé

1 January 2019

1. Introduction

Background and Motivation

A recommendation system is a subclass of information filtering system whose goal is to predict the "rating" or "preference" a user would give to an item. In this project the items are movies.

Recommender systems are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. There are also recommender systems for experts collaborators, jokes, restaurants, garments, financial services, life insurance, romantic partners (online dating), and Twitter page. Major companies such as Amazon, Netflix and Spotify ares using recommendation systems. A strong recommendation system was of such importance that in 2006, Netflix offered a million dollar prize to anyone who could improve the effectiveness of its recommendation system by 10%.

In future this area will grow in its importance since many companies are collecting more data.

Within this project we will focus on the prediction of movie ratings provided by the 'MovieLense' data set. (10M Version)

Used DataSet

For this project a movie recommendation system is created using the 'MovieLens' dataset. This data set can be found and downloaded here:

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

Goal

The goal is to train a machine learning algorithm using the inputs of a provided subset to predict movie ratings in a provided validation set (provided by edx staff).

Furthermore visualisations of the data is necessary (using ggplot2) in order to identify factors that could affect users ratings. Four main models are then established, where their RMSE are calculated to assess the quality of the model. Finally, we apply the best model to the provided validation set and submitt our predictions.

Read in of Data

The raw data has to be downloaded from the provided link above.

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

```
## Loading required package: tidyverse
## -- Attaching packages -------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.0     v forcats 0.3.0
```

```
## Warning: package 'ggplot2' was built under R version 3.5.1
```

```
## Warning: package 'tidyr' was built under R version 3.5.1
```

```
## Warning: package 'readr' was built under R version 3.5.1
```

```
## Warning: package 'dplyr' was built under R version 3.5.1
```

```
## -- 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
```

```
## Warning: package 'caret' was built under R version 3.5.1
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

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

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

ratings <- read.table(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")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

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

2. Method/Analysis

Data Prepatation/Cleaning

RMSE

The root-mean-square deviation (RMSD) or root-mean-square error (RMSE) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed. In this project the RMSE value is used as to main indicator reflecting the quality of the underlying model.

```
# function to calcualte the RMSE values
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2,na.rm = T))
}
```

Split data: Test and Train Data

An algorithm will be build in order to predict users ratings for movies they had not seen yet. The MovieLens datset is split into two different sets: -the data set for building our algorithm (called: edx) and -the data set for testing (called: validation). The validation set will be 10% of the MovieLens data.

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

validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

removed <- anti_join(temp, validation)
```

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

```
edx <- rbind(edx, removed)
validation_CH <- validation
validation <- validation %>% select(-rating)
```

Adding year and genre

In the original data set the information of the release year is also available, however it is hidden in the title column in brackets. Thus the release year is transformed in a seperate column. This is necessary to use dependecies between release year and rating. Additionally for each movie there are multiple genres highlighted. Here its necessary to split up the genres in seperate rows.

```
# Data preparation: Modify the edx/validation data set so that we have year as a column as well
edx <- edx %>%
  mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>%
  mutate(year = as.numeric(str_sub(title,-5,-2)))
```

Data Exploration/Visualisation

General data frame information

```r
nrow(edx)
```

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

```r
ncol(edx)
```

```
## [1] 7
```

```r
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 year
## 1                Comedy|Romance 1992
## 2         Action|Crime|Thriller 1995
## 3  Action|Drama|Sci-Fi|Thriller 1995
## 4       Action|Adventure|Sci-Fi 1994
## 5 Action|Adventure|Drama|Sci-Fi 1994
## 6       Children|Comedy|Fantasy 1994
```

```r
#How many different movies
n_distinct(edx$movieId)
```

```
## [1] 10677
```

```r
#How many different user
n_distinct(edx$userId)
```

```
## [1] 69878
```

```r
#How many different genres (and combination)
n_distinct(edx$genres)
```

```
## [1] 797
```

```r
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               year
##  Length:9000055     Length:9000055     Min.   :1915
##  Class :character   Class :character   1st Qu.:1987
##  Mode  :character   Mode  :character   Median :1994
##                                        Mean   :1990
##                                        3rd Qu.:1998
##                                        Max.   :2008
```

Distribution of the ratings

The distribution of the ratings shows that users tend to rate movies rather higher than lower. Most of the users rate the movies between 3 and 4. However the maxium rating of 5 is also very common. Ratings lower than 2 are rather rare. Additionally the majority of the users try to avoid an in-between ranking (0.5, 1.5, 2.5, 3.5, 4.5).

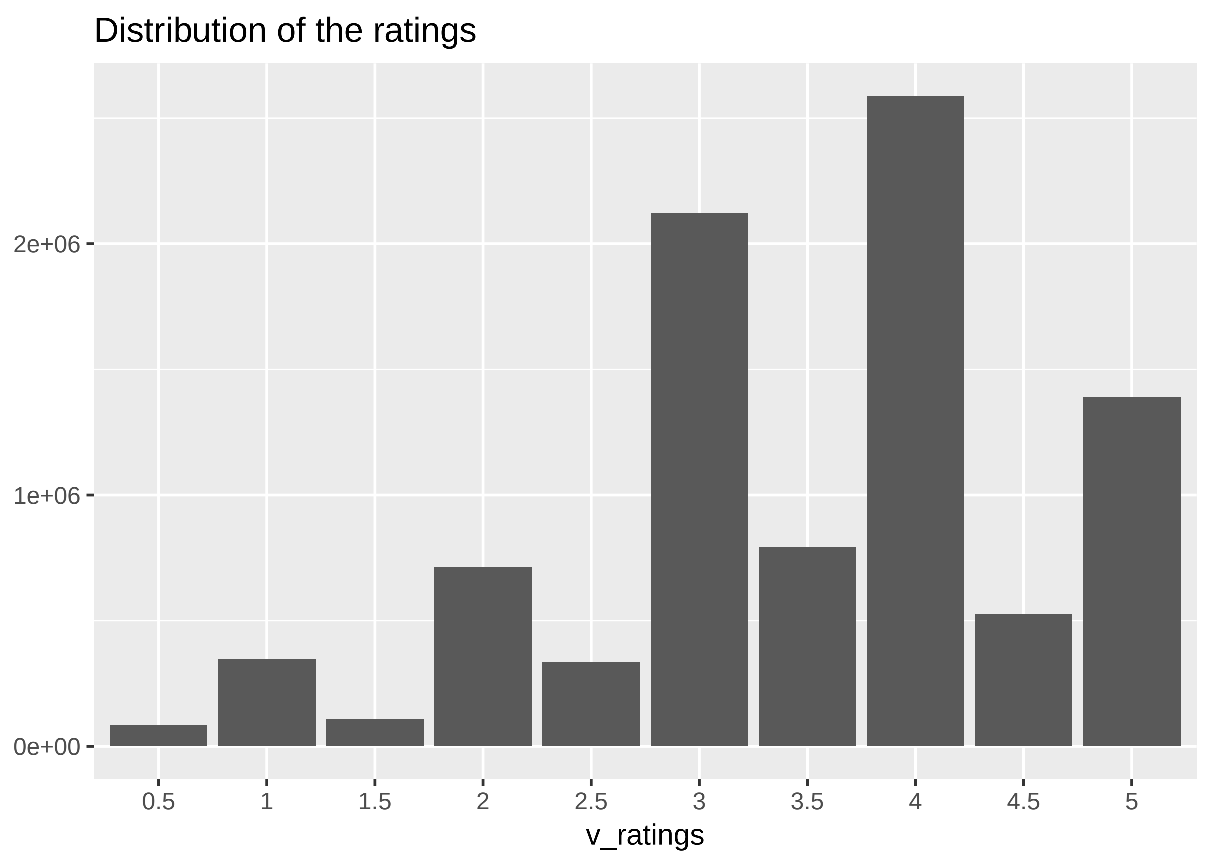
```
v_ratings <- as.vector(edx$rating)
unique(v_ratings)
```

```
##  [1] 5.0 3.0 2.0 4.0 4.5 3.5 1.0 1.5 2.5 0.5
```

```
table_ratings <- table(v_ratings)
table_ratings
```

```
## v_ratings
##     0.5       1     1.5       2     2.5       3     3.5       4     4.5
##   85374  345679  106426  711422  333010 2121240  791624 2588430  526736
##       5
## 1390114
```

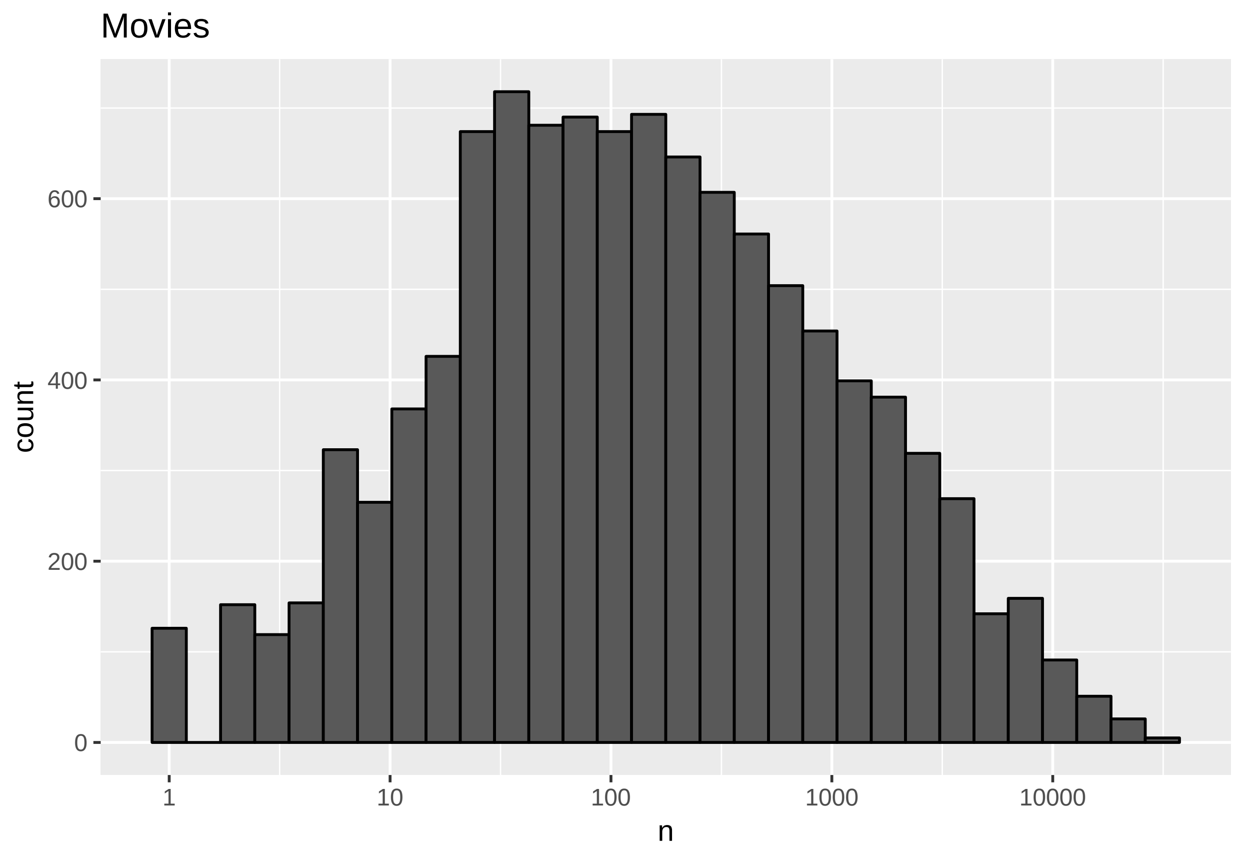
```
v_ratings <- v_ratings[v_ratings != 0]
v_ratings <- factor(v_ratings)
qplot(v_ratings) +
  ggtitle("Distribution of the ratings")
```



How often a single movie has been rated?

We can see that most movies have been reviewed between 50 and 1000 times. What has to be pointed out is that around 125 movies have been rated only once, which will be challenging to predict. A way how to adjust for this relation is through regularization, which will be included later on in the used models. A penalty term is included in this model, which becomes more prominent as the sample size decreases.

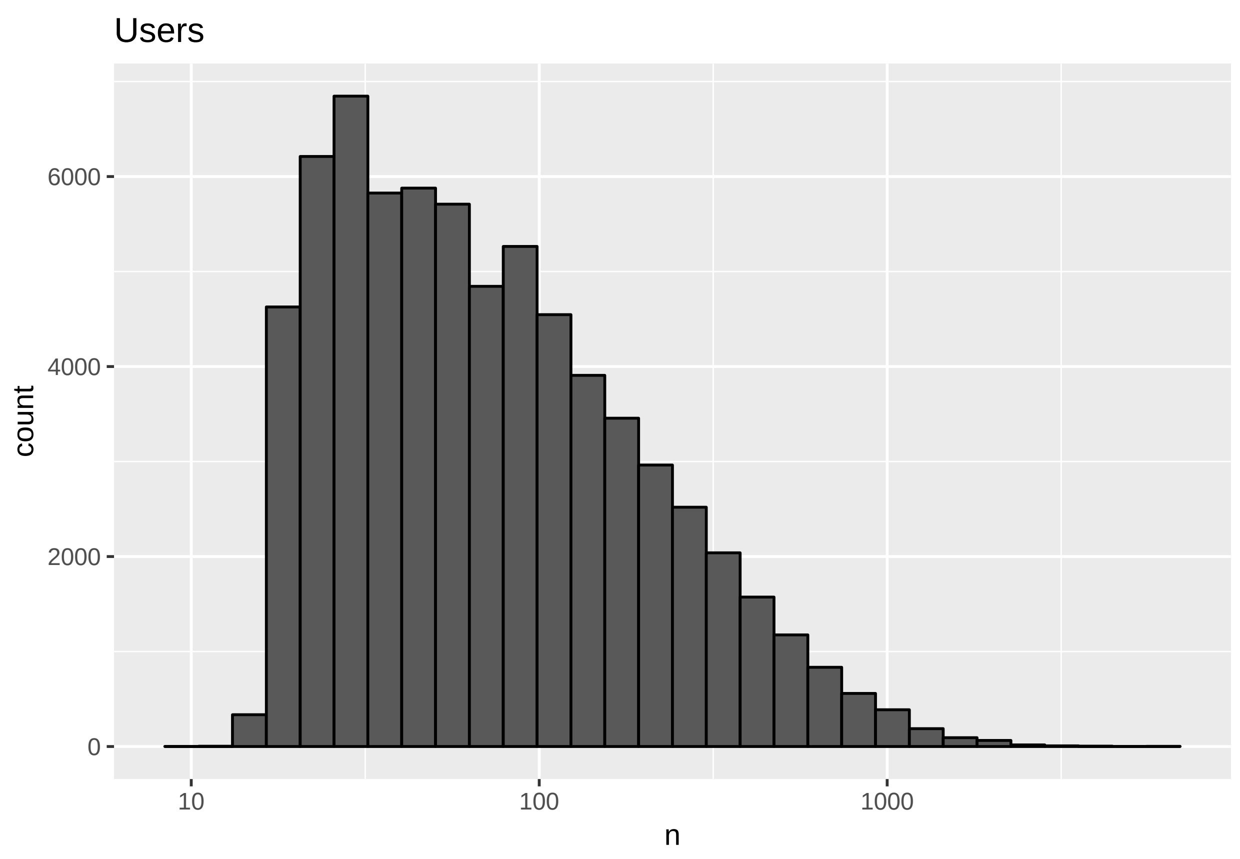
```r
edx %>% count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Movies")
```



How often a single user did review?

Most of the users reviewed fewer than 100 movies, but more than 30. Here later on a penalty term will be included in the model as well, similar to above.

```r
edx %>% count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "black") +
  scale_x_log10() +
  ggtitle("Users")
```



Release year vs rating

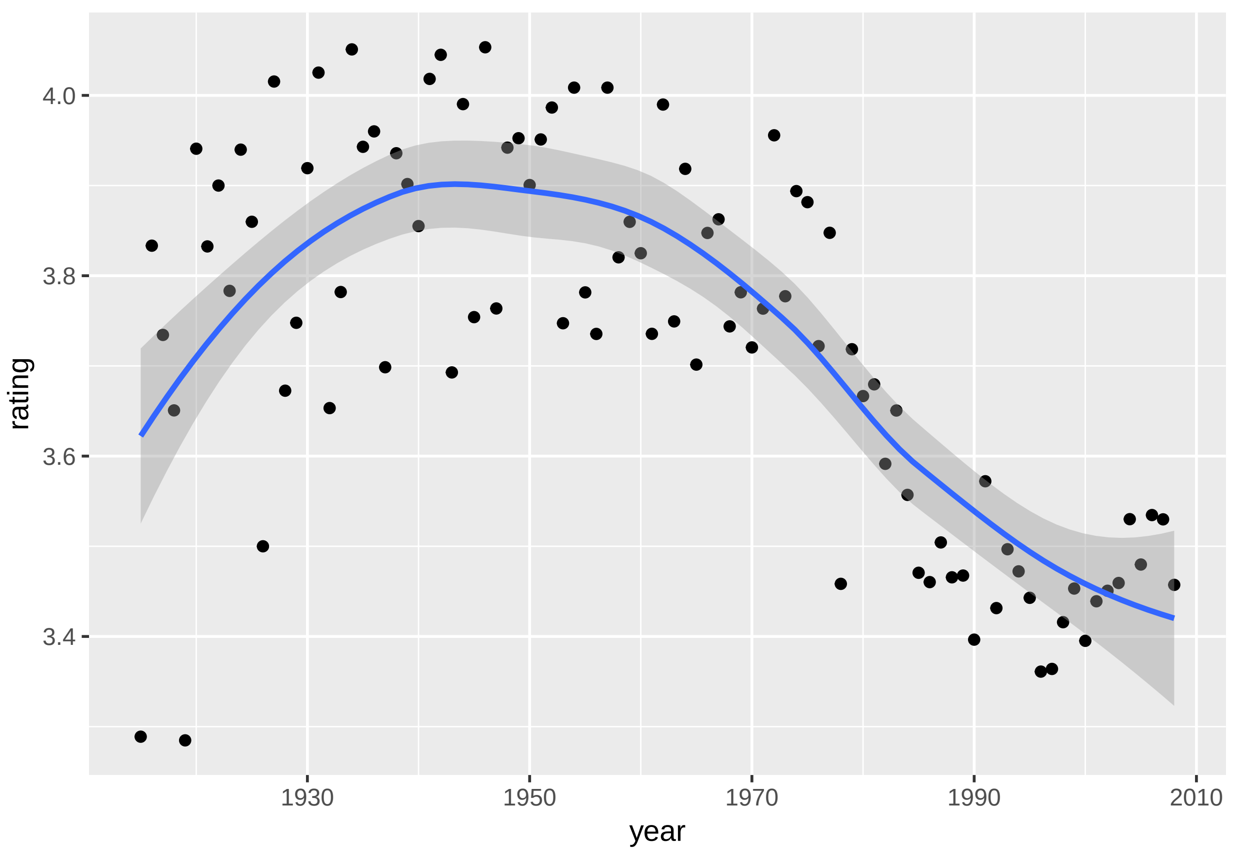
You might think that the release year has very little effect on the rating, however there is a trend observable. The most recent years have in average lower rating than in the earlier years.Hypothesis: We will not take into account this aspect because of R Limitation with big dataset.

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```r
edx %>% group_by(year) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(year, rating)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Genre vs rating

We can also suspect that genres might have certain effects. Hypothesis: We will not take into account this aspect because of R Limitation with big dataset.

Data Analysis/Model Preparation

```r
# Initialize docuemntation of RMSE for comparison purpose
rmse_results <- data_frame()
```

sample estimate- mean

First we calculate the mean rating for our data set. The expected rating of the underlying data set is between 3 and 4.

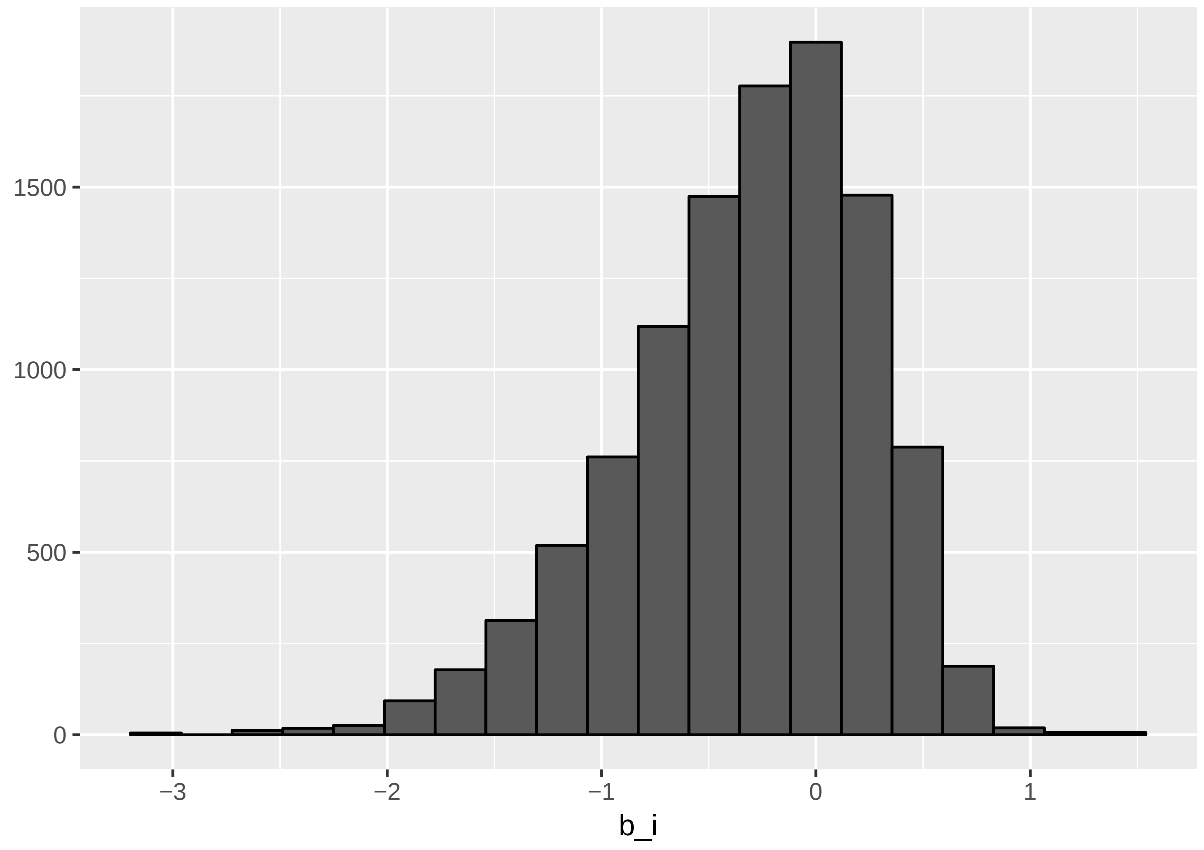
```r
mu <- mean(edx$rating)
mu
```

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

Penalty term - Movie effect

Intuitivly good movies are more likely to get good review, and vice versa for bad. the movie effect should account for this. The histogram is skewed left, meaning that more movies have negative effects.

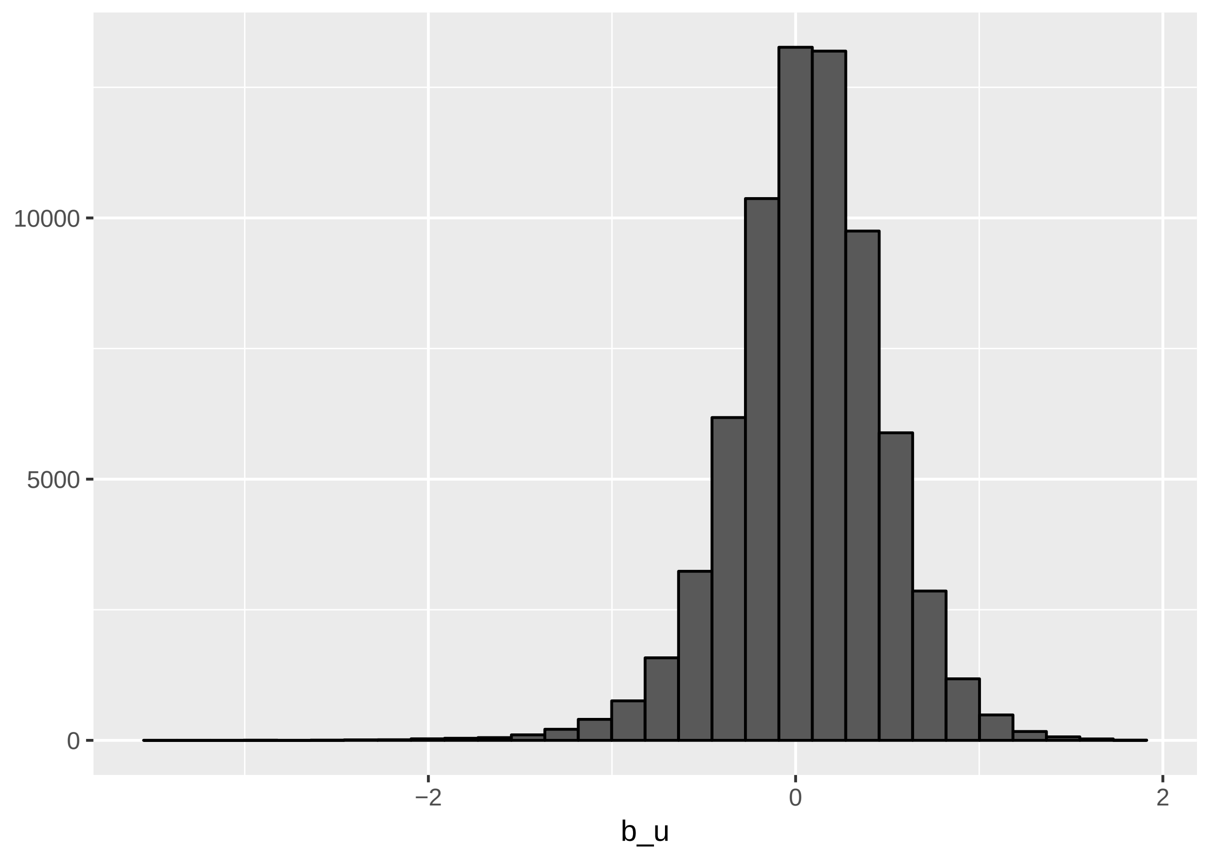
```r
movie_avgs_norm <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs_norm %>% qplot(b_i, geom ="histogram", bins = 20, data = ., color = I("black"))
```



Penalty term - User effect

Obviously users can have the same effect as mentioned above: some users tend to give lower rankings than others and vice versa.

```r
user_avgs_norm <- edx %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
user_avgs_norm %>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"))
```



Model Creation

The quality of the model will be measured by RMSE. The lower the better.

Naive Model

The first amd simpliest attempt is to create a prediction system that only uses the sample mean. This means every prediction is the sample average. Unsurprisingly the RMSE is quite high.

```
# Naive Model -- mean only
naive_rmse <- RMSE(validation_CH$rating,mu)
## Test our results based on the simple prediction
naive_rmse
```

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

```
## See result
rmse_results <- data_frame(method = "Using mean only", RMSE = naive_rmse)
rmse_results
```

```
## # A tibble: 1 x 2
##   method           RMSE
##   <chr>           <dbl>
## 1 Using mean only  1.06
## Store prediction in data frame
```

Movie Effect model

By adding the the movie effect the RMSE could be improved.

```r
#Movie effects only
predicted_ratings_movie_norm <- validation %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  mutate(pred = mu + b_i)
model_1_rmse <- RMSE(validation_CH$rating,predicted_ratings_movie_norm$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087

```r
rmse_results
```

```
## # A tibble: 2 x 2
##   method               RMSE
##   <chr>               <dbl>
## 1 Using mean only    1.06
## 2 Movie Effect Model 0.944
```

Movie and User Effect Model

By adding the user effect the RMSE improves even further

```r
predicted_ratings_user_norm <- validation %>%
  left_join(movie_avgs_norm, by='movieId') %>%
  left_join(user_avgs_norm, by='userId') %>%
  mutate(pred = mu + b_i + b_u)
## Use test set, then join movie averages and user averages, and find prediction
## Prediction is mean plus user effect plus movie effect
model_2_rmse <- RMSE(validation_CH$rating,predicted_ratings_user_norm$pred)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and User Effect Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488

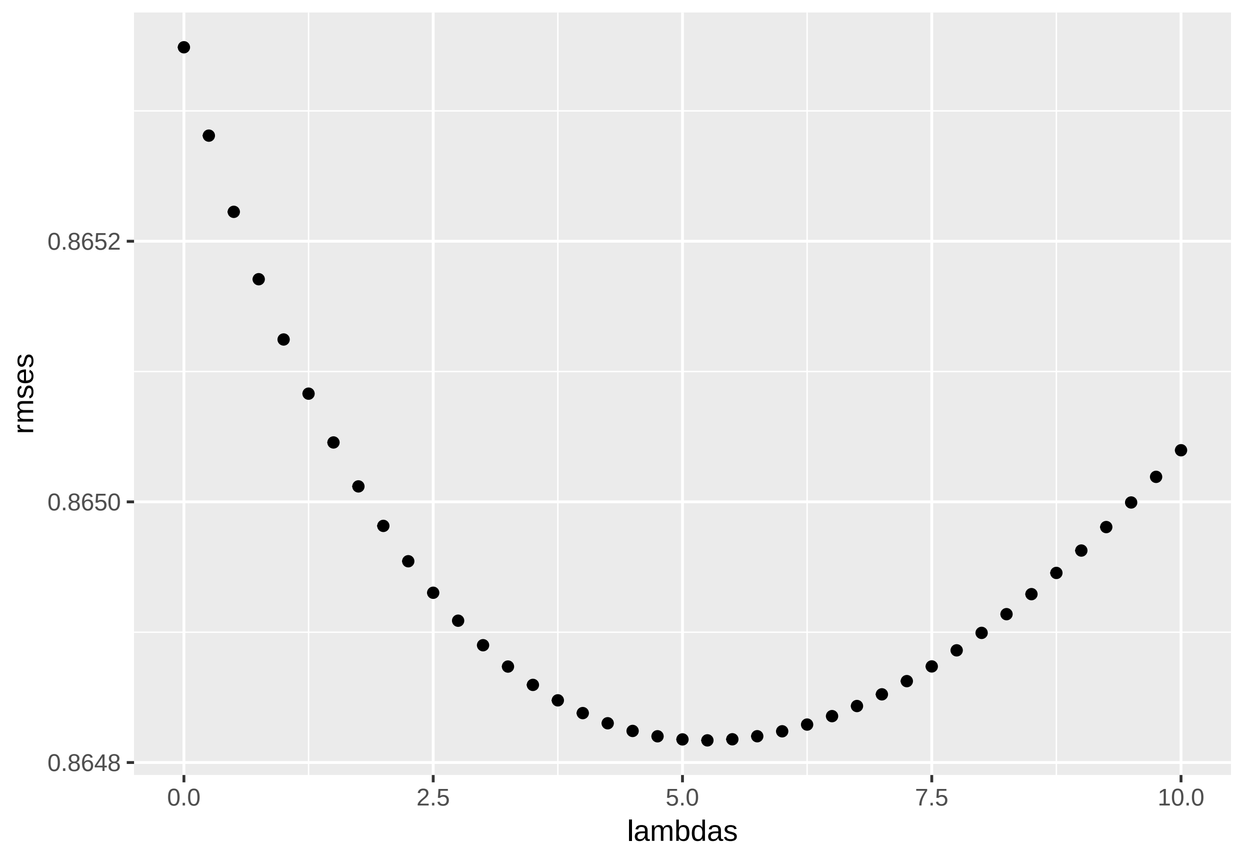
```
rmse_results
```

```
## # A tibble: 3 x 2
##   method                       RMSE
##   <chr>                       <dbl>
## 1 Using mean only             1.06
## 2 Movie Effect Model          0.944
## 3 Movie and User Effect Model 0.865
```

Regularized Movie and User Model

Here we use the concept of regularization in order to account fo the effect of low numbers of ratings both for movies and users. As a reminder, we saw before that a few movies very rated only once and some users only rated very few movies. This fact can influence the prediction strongly. Regularization is a method to reduce the effect of overfitting.

```
lambdas <- seq(0, 10, 0.25)
# Sequence of lambdas to use
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

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

  predicted_ratings <- validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred

  return(RMSE(validation_CH$rating,predicted_ratings))
})
## For each lambda, we find the b_i and the b_u, then make our prediction and test.
qplot(lambdas, rmses)
```



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

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

```r
## Plot the lambdas vs the rmses, see which has the best accuracy, and choose that for lambda.
movie_avgs_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())
## Using lambda, find the movie effects
user_avgs_reg <- edx %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda), n_u = n())
## Using lambda, find the user effects
predicted_ratings_reg <- validation %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  left_join(user_avgs_reg, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
## Make our predicted ratings
model_3_rmse <- RMSE(validation_CH$rating,predicted_ratings_reg)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie and User Effect Model",
                                     RMSE = model_3_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Using mean only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie and User Effect Model	0.8648170

```
rmse_results
```

```
## # A tibble: 4 x 2
##   method                                   RMSE
##   <chr>                                   <dbl>
## 1 Using mean only                         1.06
## 2 Movie Effect Model                      0.944
## 3 Movie and User Effect Model             0.865
## 4 Regularized Movie and User Effect Model 0.865
```

3. Result

RMSE overview

This are the RMSE values for the used models:

```
rmse_results %>% knitr::kable()
```

method
Using mean only
Movie Effect Model
Movie and User Effect Model
Regularized Movie and User Effect Model
Prediction rating with model
Since the model that put into account the   effects of movie, user had the best RMSE result, it will be taken as the model f

```
lambda <- 5.25
## From model 3
## Plot the lambdas vs the rmses, see which has the best accuracy, and choose that for lambda.
movie_avgs_reg <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())
## Using lambda, find the movie effects
user_avgs_reg <- edx %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+lambda), n_u = n())
## Using lambda, find the user effects
predicted_ratings <- validation %>%
  left_join(movie_avgs_reg, by='movieId') %>%
  left_join(user_avgs_reg, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>% group_by(userId,movieId) %>%
  summarize(pred_2 = mean(pred))
```

Accuracy

Before to the prediction can be used it has to be slightly modified. Since only a certain numbers are allowed (0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5) but the outputs of the models are continous, the final prediction output has to be rounded. Additionally the values of 0 and greater than 5 are not allowed per definition. Those values have to be replaced.

The accuracy is roughly 25%. However the “close accuracy” is roughly 65% (it tells us how often we were within 0.5 stars of the actual rating).

```
## Round our predicted_ratings
predicted_ratings$pred_2 <- round(predicted_ratings$pred_2*2)/2
## Make sure all ratings are between 0.5 and 5
predicted_ratings$pred_2[which(predicted_ratings$pred_2<1)] <- 0.5
predicted_ratings$pred_2[which(predicted_ratings$pred_2>5)] <- 5

## See direct accuracy
mean(predicted_ratings$pred_2 == validation_CH$rating)
```

```
## [1] 0.2480012
```

```
## See close accuracy--within 0.5 stars
x <- sum(predicted_ratings$pred_2 <= validation_CH$rating + 0.5 & predicted_ratings$pred_2 >= validatio
x/length(predicted_ratings$pred_2)
```

```
## [1] 0.6439086
```

4. Conclusion

Based on the RMSE values the best model with this submission project is the regularized model including the effect of movie and user

The accuracy was rather low (25%), which can be explained in many ways. First of all we did not use a categorical predictor, instead we used a numerical method. This was probably not a good choice, however it was suggested by the edx course and it was even used for the NEtflix challenge. There however the measurement of model quality was the RMSE value and not the accuracy.(the winner had a RMSE of 0,857) Additionally the effect that certains users prefer certain genres was not taken into account. The gender could also play an important role, e.g. female users prefer romances over war movies, etc.. In this sense age might also play an important role, e.g. older people do not like animation movies. All those factors were not taken into account. To make an even better model, we could have done web scrapping by linking IMDB to each movie, adding the columns of lead actor, directors.

The close accuracy for being within 0.5 stars is ok (65%).