HarvardX: PH125.9x Data Science Capstone-MovieLens Rating Prediction Project

Saurav

Chatterjee

MD

November 24,

2020

Contents

1 Overview

This project is related to the MovieLens Project of the HarvardX: PH125.9x Data Science: Capstone course. The MovieLens data set was collected by GroupLens Research. For the project. attempt will be made to predict movie ratings based on variables available in the dataset. The given dataset will be prepared and set up for data analysis-including separating the data into a training and validation subsets. An exploratory data analysis will be carried out in the training dataset in order to develop a machine learning algorithm that could predict movie ratings, and an optimal model fit will be ascertained with appropriate explanations. Finally, the report will outline some limitations and concluding remarks.

1.1 Introduction

For this project we will focus on creating a movie recommendation system using the 10M version of MovieLens dataset, collected by GroupLens Research.

1.2 Aim of the project

The aim in this project is to train a machine learning algorithm that predicts user ratings (from 0.5 to 5 stars) using the inputs of a provided training dataset (edx dataset provided by the staff) to predict movie ratings in a provided validation set.

The value used to evaluate algorithm performance is the Root Mean Square Error, or RMSE. RMSE is one of the most used measure of the differences between values predicted by a model and the values observed. RMSE is a measure of accuracy, to compare forecasting errors of different models for a particular dataset, a lower RMSE is better than a higher one. The effect of each error on RMSE is proportional to the size of the squared error; thus larger errors have a disproportionately large effect on RMSE. Consequently, RMSE is sensitive to outliers. Finally, the best resulting model will be used to predict the movie ratings in the validation dataset.

1.3 Dataset

The MovieLens dataset is automatically downloaded

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

```
############################################################
# Create edx set, validation set, and submission file
############################################################
#Create test and validation sets
# Create edx set, validation set, and submission file
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-
project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-
project.org")
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")
```

In order to predict in the most accurate way possible, the movie rating of the users that haven't seen the movie yet, the MovieLens dataset will be split into 2 subsets that will be the "edx", a training subset to train the algorithm, and "validation" a subset to test the movie ratings.

```
# The Validation subset will be 10% of the MovieLens data.

set.seed(123)

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 subset:

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)
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Algorithm development is to be carried out on the "edx" subset only, as "validation" subset will be used to test the final algorithm.

2 Methods and Analysis

2.1 Data Exploration and Analysis

```
## the dataset and its basic summary statistics

# intial 7 rows with header

head(edx)

# basic summary statistics

summary(edx)

#  total number of observations

tot_observation <- length(edx$rating) + length(validation$rating)
```

To get familiar with the dataset, we find the first rows of "edx" subset as below. The subset contain the six variables "userID", "movieID", "rating", "timestamp", "title", and "genres". Each row represents a single rating of a user for a single movie.

```
##    userId  movieId  rating timestamp                 title
## 1       1      122       5 838985046      Boomerang (1992)
## 2       1      185       5 838983525          Net, The (1995)
## 4       1      292       5 838983421      Outbreak (1995)
## 5       1      316       5 838983392      Stargate (1994)
## 6       1      329       5 838983392 Star Trek: Generations (1994)
## 7       1      355       5 838984474 Flintstones, The (1994)
##                                  genres
## 1                       Comedy|Romance
## 2                Action|Crime|Thriller
## 4   Action|Drama|Sci-Fi|Thriller
## 5             Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7             Children|Comedy|Fantasy
```

A summary of the subset confirms that there are no missing values.

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

Data pre-processing and exploratory analysis

```
#  Since RMSE (root mean squre error) is used frequently so we define a function
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings-predicted_ratings)^2,na.rm=T))
}
# Modify the columns to suitable formats that can be further used for analysis
# Modify the year as a column in the edx & validation datasets
edx <- edx %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation <- validation %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
validation_CM <- validation_CM %>% mutate(year = as.numeric(str_sub(title,-5,-2)))
# Modify the genres variable in the edx & validation dataset (column separated)
split_edx  <- edx  %>% separate_rows(genres, sep = "\\|")
split_valid <- validation   %>% separate_rows(genres, sep = "\\|")
split_valid_CM <- validation_CM  %>% separate_rows(genres, sep = "\\|")

# The 1st rows of the edx & split_edx datasets are presented below:
head(edx)
head(split_edx)
# edx Summary Statistics
summary(edx)
# Number of unique movies and users in the edx dataset
edx %>% summarize(n_users = n_distinct(userId), n_movies = n_distinct(movieId))
```

Total movie ratings per genre

```
genre_rating <- split_edx%>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

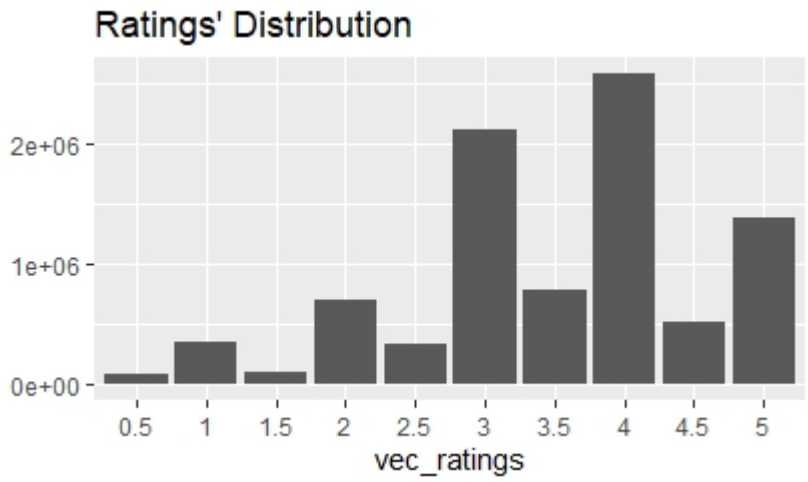
Ratings distribution

```
vec_ratings <- as.vector(edx$rating)
unique(vec_ratings)
vec_ratings <- vec_ratings[vec_ratings != 0]
vec_ratings <- factor(vec_ratings)
qplot(vec_ratings) +
  ggtitle("Ratings' Distribution")
```

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

Users have a preference to rate movies rather higher than lower as shown by the distribution of ratings below. 4 is the most common rating, followed by 3 and 5. 0.5 is the least common rating. In general, half rating are less common than whole star ratings.

Rating distribution

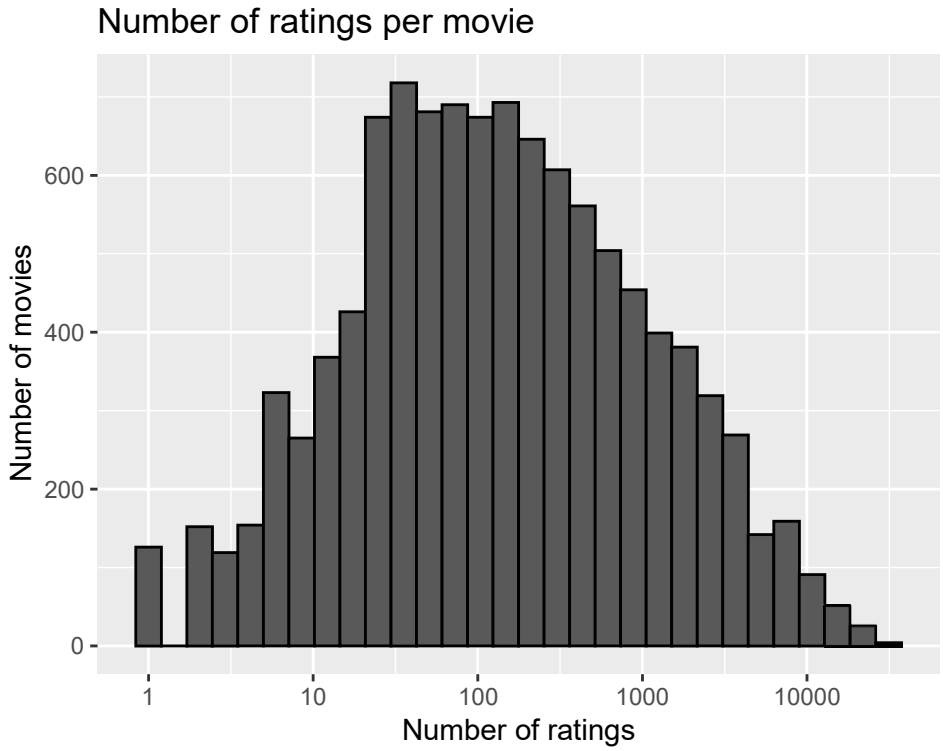


Data Analysis Strategies

- Some movies are rated more often than others (e.g. blockbusters are rated higher). How to incorporate this in our model: find movie bias.
- Some users are positive and some have negative reviews because of their own personal liking/disliking regardless of movie. How to address this characteristics: find users bias.
- The popularity of the movie genre depends strongly on the contemporary issues. So we should also explore the time dependent analysis. How to approach this idea: find the genre popularity over the years
- Do the users mindset also evolve over time? This can also effect the average rating of movies over the years. How do visualize such effect: plot rating vs release year

Thus regularization and a penalty term will be applied to the models in this project. Regularizations are techniques used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting (the production of an analysis that corresponds too closely or exactly to a particular set of data, and may therefore fail to fit additional data or predict future observations reliably). Regularization is a technique used for tuning the function by adding an additional penalty term in the error function. The additional term controls the excessively fluctuating function such that the coefficients don't take extreme values.

```
edx %>% count(movieId)
%>% ggplot(aes(n)) +
geom_histogram(bins = 30, color = "black") +
scale_x_log10() +
xlab("Number of ratings") +
   ylab("Number of movies") +
ggtitle("Number of ratings per movie")
```

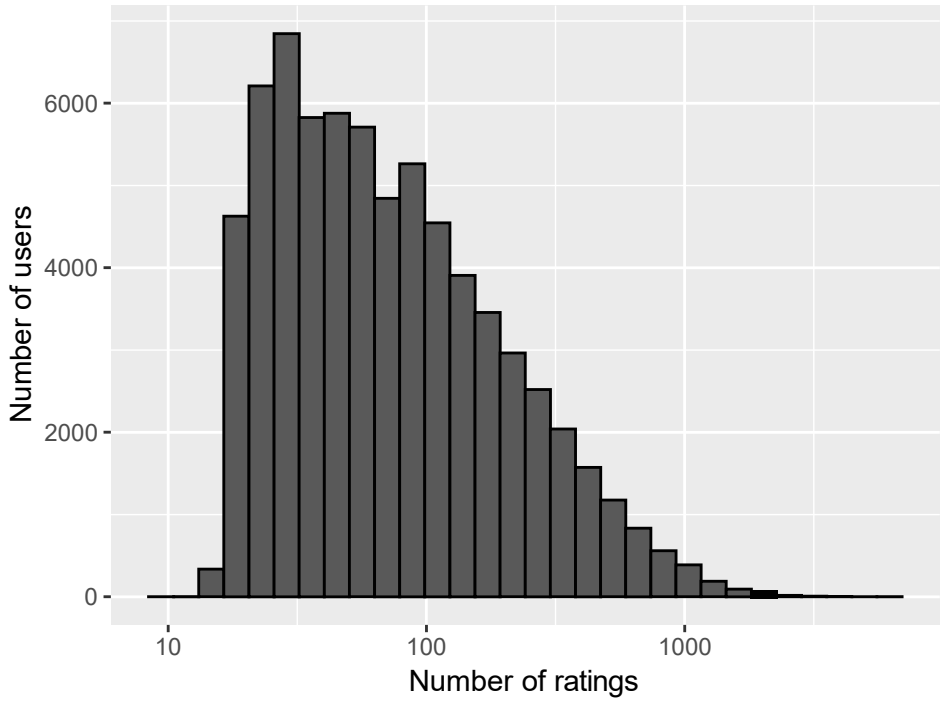


The histogram shows some movies have been rated very few number of times. So they should be given lower importance in movie prediction.

```
edx %>%
   group_by(movieId) %>%
   summarize(count = n()) %>%
   filter(count == 1) %>% left_join(edx,
   by = "movieId") %>% group_by(title)
   %>%
   summarize(rating = rating, n_rating = count) %>%
   slice(1:20)  %>%
   knitr::kable()
```

Number of ratings given by users

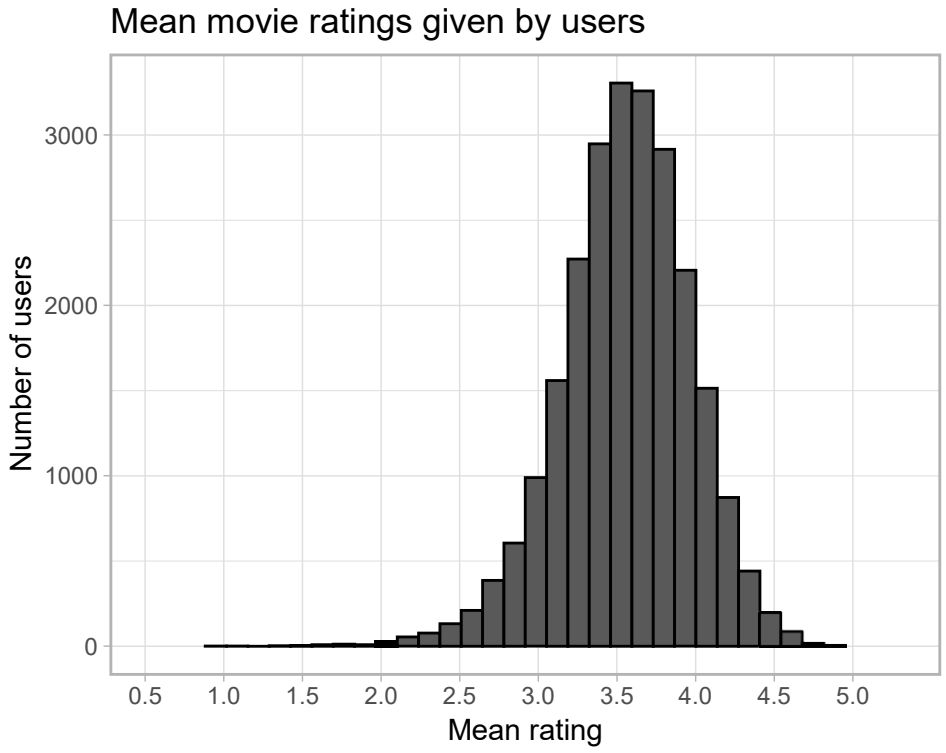
```
edx %>%
count(userId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "black") +
scale_x_log10() +
xlab("Number of ratings") +
ylab("Number of users") +
ggtitle("Number of ratings given by users")
```



We can observe that the majority of users have rated between 30 and 100 movies. So, a user penalty term need to be included later in our models. Furthermore, users differ vastly in how critical they are with their ratings. Some users tend to give much lower star ratings and some users tend to give higher star ratings than average. The visualization below includes only users that have rated at least 100 movies.

```
edx %>% group_by(userId)
  %>%
  filter(n() >= 100) %>% summarize(b_u
  = mean(rating)) %>% ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black") +
  xlab("Mean rating") +
  ylab("Number of users") +
```

```
  ggtitle("Mean movie ratings given by users") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  theme_light()
```



2.2 Modelling Approach

We write now the loss-function, previously anticipated, that compute the RMSE, defined as follows:

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

with N being the number of user/movie combinations and the sum occurring over all these combinations. The RMSE is our measure of model accuracy. We can interpret the RMSE similarly to a standard deviation: it is the typical error we make when predicting a movie rating. If its result is larger than 1, it means that our typical error is larger than one star, which is not a good result. The written function to compute the RMSE for vectors of ratings and their corresponding predictions is:

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

The lower the better, as said previously.

2.2.1 I. Average movie rating model

The first basic model predicts the same rating for all movies, so we compute the dataset's mean rating. The expected rating of the underlying data set is between 3 and 4. We start by building the simplest possible

recommender system by predicting the same rating for all movies regardless of user who give it. A model based approach assumes the same rating for all movie with all differences explained by random variation :

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

with $E_{u,i}$ independent error sample from the same distribution centered at 0 and μ the "true" rating for all movies. This very simple model makes the assumption that all differences in movie ratings are explained by random variation alone. We know that the estimate that minimize the RMSE is the least square estimate of $Y_{u,i}$, in this case, is the average of all ratings: The expected rating of the underlying data set is between 3 and 4.

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

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

If we predict all unknown ratings with μ or mu, we obtain the first naive RMSE:

```
naive_rmse <- RMSE(validation$rating, mu)
naive_rmse
```

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

Here, we represent results table with the first RMSE:

```
rmse_results <- data_frame(method = "Average movie rating model", RMSE = naive_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.061202

This give us our baseline RMSE to compare with next modelling approaches.

In order to do better than simply predicting the average rating, we incorporate some of insights we gained during the exploratory data analysis.

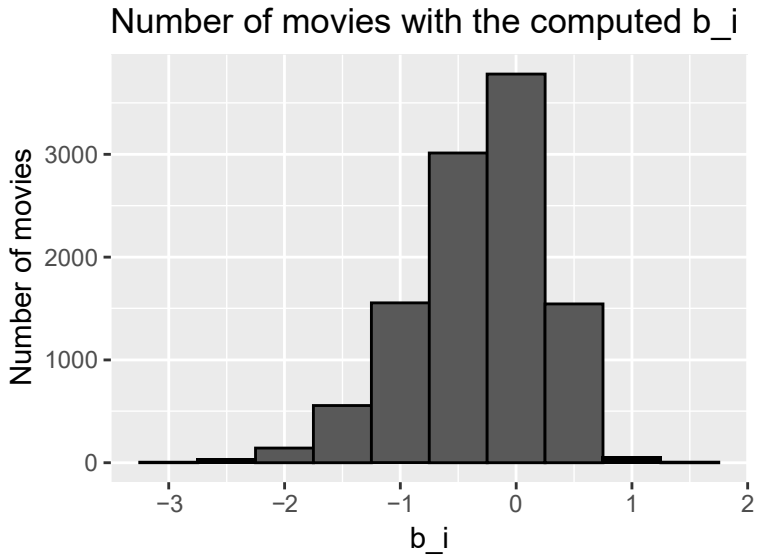
2.2.2 II. Movie effect model

To improve above model we focus on the fact that, from experience, we know that some movies are just generally rated higher than others. Higher ratings are mostly linked to popular movies among users and the opposite is true for unpopular movies. We compute the estimated deviation of each movie's mean rating from the total mean of all movies μ. The resulting variable is called "b" (as bias) for each movie "i" b_i, that represents average ranking for movie i:

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

The histogram is left skewed, implying that more movies have negative effects

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"),
ylab = "Number of movies", main = "Number of movies with the computed b_i")
```



This is called the penalty term movie effect.

Our prediction improve once we predict using this model.

```
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie effect model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087

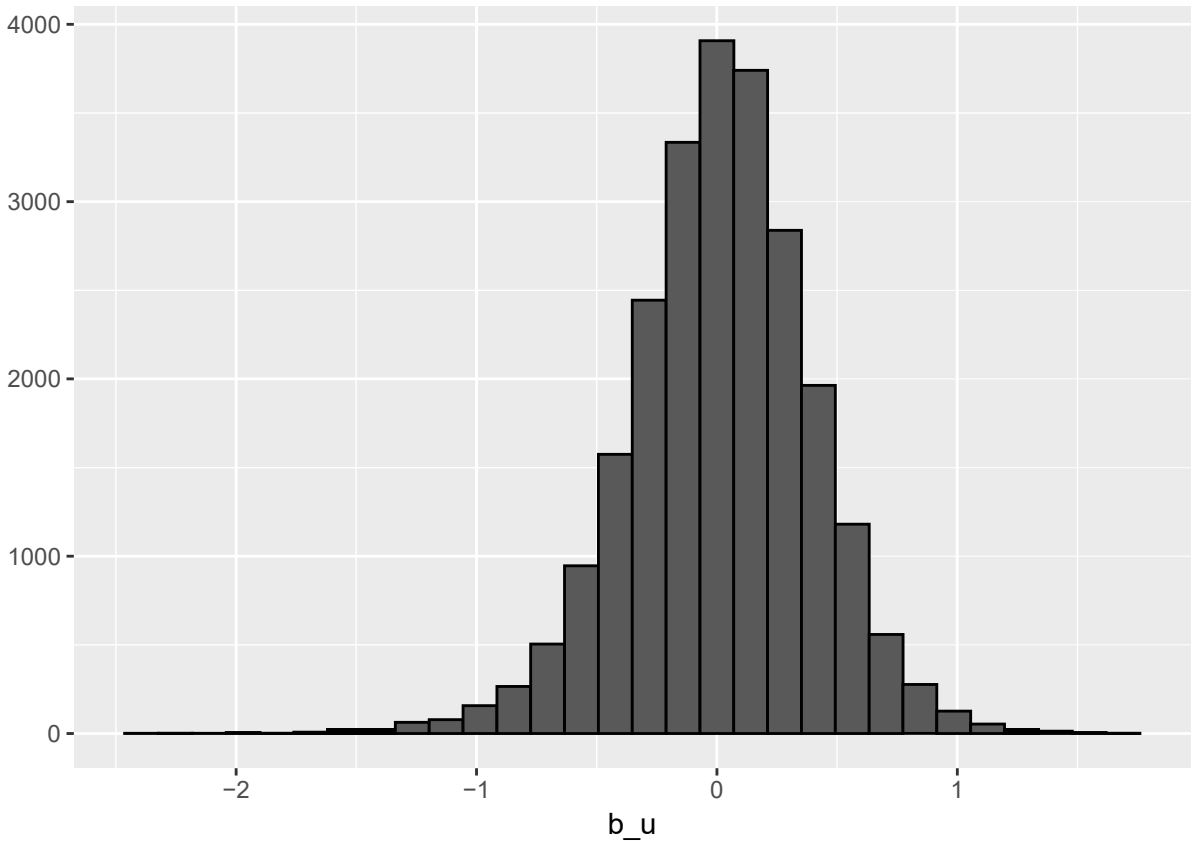
So we have predicted movie rating based on the fact that movies are rated differently by adding the computed b_i to μ. If an individual movie is on average rated worse that the average rating of all movies μ , we predict that it will rated lower that μ by b_i, the difference of the individual movie average from the total average.

We can see an improvement but this model does not consider the individual user rating effect.

2.2.3 III. Movie and user effect model

We compute the average rating for user μ, for those that have rated over 100 movies, said penalty term user effect. Users may affect the ratings positively or negatively.

```
user_avgs<- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating - mu - b_i))
user_avgs%>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"))
```



There is substantial variability across users as well: some users are very picky and others love every movie. This implies that further improvement to our model may be:

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

where b_u is a user-specific effect. If a cranky user (negative b_u rates a great movie (positive b_i), the effects counter each other and we may be able to correctly predict that this user gave this great movie a 3 rather than a 5.

We compute an approximation by computing μ and b_i, and estimating b_u, as the average of

$$Y_{u,i} - \mu - b_i$$

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

We can now construct predictors and see RMSE improves:

```
predicted_ratings <- validation%>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and user effect model",
```

```
                                     RMSE = model_2_rmse))
rmse_results %>% knitr::kable()
```

Method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488

Our rating predictions further reduced the RMSE. But the supposedly "best " and "worst "movie were rated by few users, in most cases just one user. These movies were mostly obscure ones. This is because with a few users, we have more uncertainty. Therefore, larger estimates of b_i, negative or positive, are more likely. Large errors can increase our RMSE.

In our data exploration, some users are more actively participated in movie reviewing. There are also users who have rated very few movies (less than 30 movies). On the other hand, some movies are rated very few times (say 1 or 2). These are basically noisy estimates that we should not trust. Additionally, RMSE are sensitive to large errors. Large errors can increase our residual mean squared error. So we must put a penalty term to give less importance to such effect.

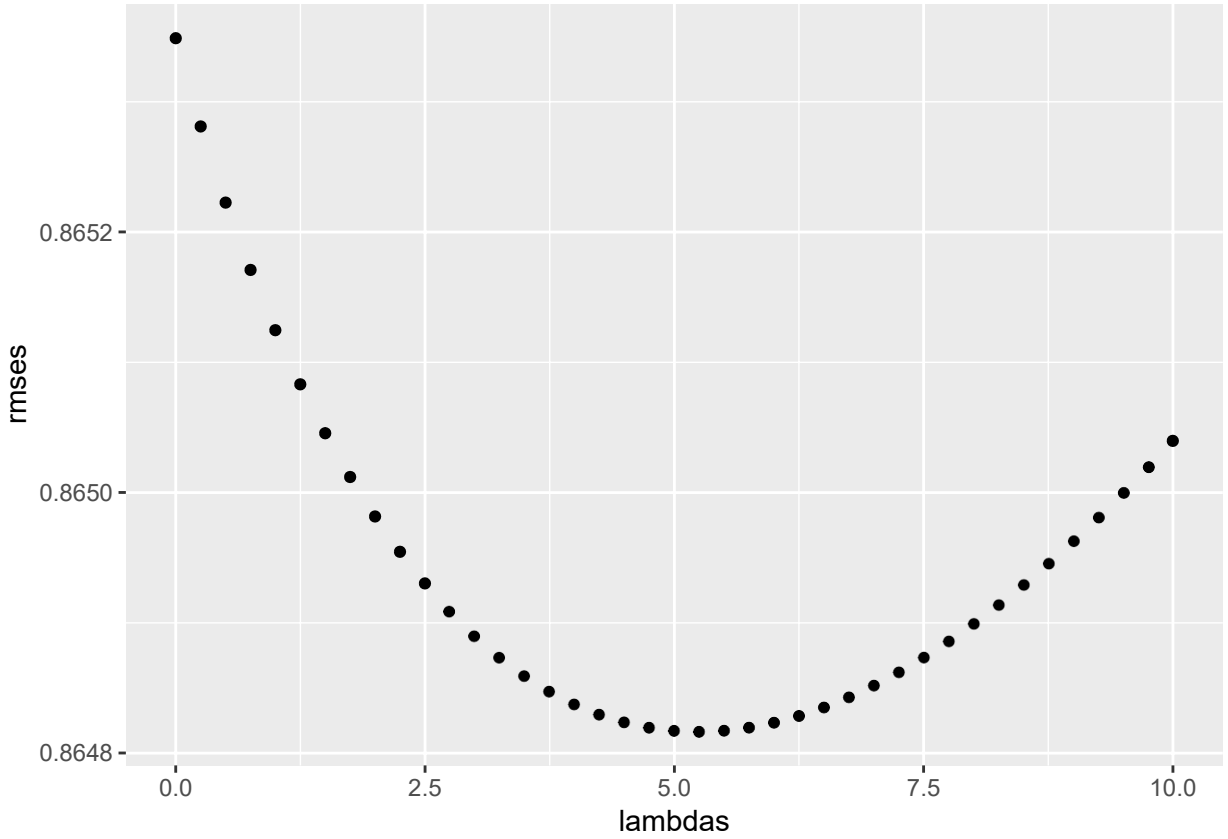
2.2.4 IV. Regularized movie and user effect model

So estimates of b_i and b_u are caused by movies with very few ratings and in some users that only rated a very small number of movies. Hence this can strongly influence the prediction. The use of the regularization permits to penalize these aspects. We should find the value of lambda (that is a tuning parameter) that will minimize the RMSE. This shrinks the b_i and b_u in case of small number of ratings.

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

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) %>%
    pull(pred)

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

We plot RMSE vs lambdas to select the optimal lambda

```
qplot(lambdas, rmses)
```



For the full model, the optimal lambda is:

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

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

For the full model, the optimal lambda is: 5.25

The new results will be:

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized movie and user effect model",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Regularized movie and user effect model	0.8648170

3 Results

The RMSE values of all the represented models are the following:

Method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087
Movie and user effect model	0.8653488
Regularized movie and user effect model	0.8648170

We therefore found the lowest value of RMSE that is 0.8648170 using the 'regularized' movie and user effect model.

4 Discussion

So we can confirm that the final model for our model is the following:

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

This model will work well if the average user doesn't rate a particularly good/popular movie with a large positive b_i.

5 Conclusion

A machine learning algorithm to predict movie ratings with MovieLens dataset was developed. The regularized model including the effect of user is characterized by the lower RMSE value and is hence the optimal model to use for the present project. Other available variables lead to no or insignificant improvements. The RMSE table shows an improvement of the model over different assumptions. The simplest model 'Using mean only' calculates the RMSE more than 1, which means we may miss the rating by one star. Then incorporating 'Movie effect' and 'Movie and user effect' on model provide an improvement by 5% and 13.5% respectively. This is substantial improvement given the simplicity of the model. A deeper insight into the data revealed some data point in the features have larger effects on errors. So a regularization model was used to penalize such data points to obtain significant improvement over the baseline model.