HarvardX Data Science Capstone MovieLens Project

Dushyanta Mishra

11/24/2019

Contents

Overview

This is the MovieLens Project of the HarvardX: PH125.9x Data Science: Capstone course. This report starts with a general idea of the project and its objective. Then it shows the steps involved in cleaning and prepping the data for analysis. This is followed by a detailed explanation of the methods and analysis used to develop a machine learning algorith that could predict movie ratings. Thereafter, the results are explained and the report finally ends with a few concluding remarks.

Introduction

Recommendation systems are widely used. From simple survey based product offerings to deeply complex and covenient Netflix suggestions, recommendation systems are used quite widely. These are often based on ratings that users have given to items they have previously used. Companies that sell many products to many customers and permit these customers to rate their products, like Amazon, are able to collect massive datasets that can be used to predict what rating a particular user will give to a specific item. Items for which a high rating is predicted for a given user are then recommended to that user.

The same could be done for other items, such as movies, in our case. Recommendation systems are one of the most used models in machine learning algorithms. In fact, the success of Netflix is said to be based on its strong recommendation system. The Netflix prize (open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or films), in fact, represent the high importance of algorithm for products recommendation system.

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

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 set to predict movie ratings in a provided validation set.

The measure used to evaluate the algorithm's performance is the Root Mean Square Error, or RMSE. RMSE is one of the most used measures of the differences between values predicted by a model and the observed values. 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. Four models that will be developed will be compared using their resulting RMSE in order to assess their accuracy. The evaluation criteria for this algorithm is a RMSE expected to be lower than 0.8775. The function that computes the RMSE for vectors of ratings and their corresponding predictors will be the following:

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

Finally, the best of the resulting models will be used to predict the movie ratings.

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
################################################################
# Note: this process could take a couple of minutes for loading required package: tidyverse and package
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 MovieLens dataset is split into 2 subsets, one to train the algorithm(edx) and the other to test it(validation).

```
# The Validation subset 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,]
#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
```

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

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

Methods and Analysis

Data Analysis

To get familiar with the dataset, we look at the first 6 rows of "edx" subset. The subset contains the six variables: ???userID???, ???movieID???, ???rating???, ???timestamp???, ???title???, and ???genres???. Each row represent 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
```

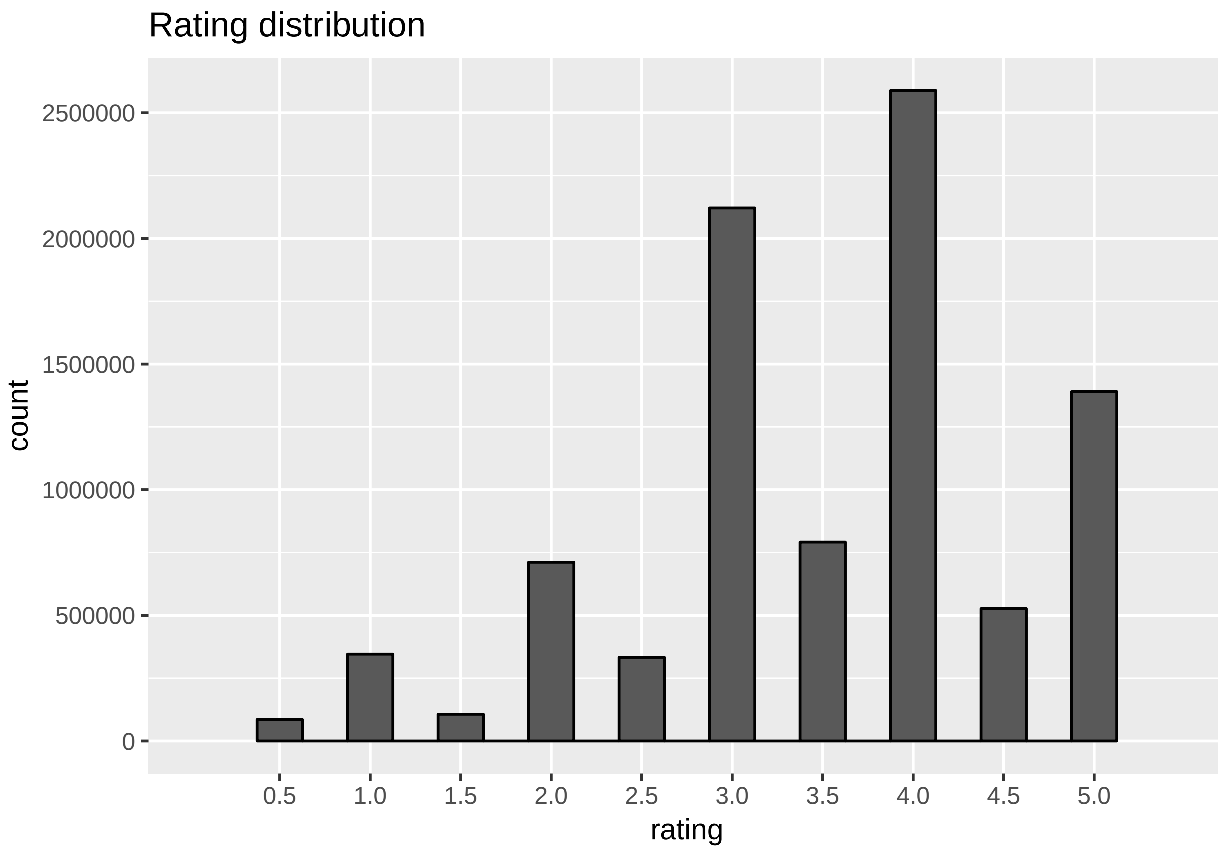
A summary of the subset confirms that there are no missing values(NAs).

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

The total no.of unique users in the edx subset is about 70,000 and it contains about 10,700 different movies:

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

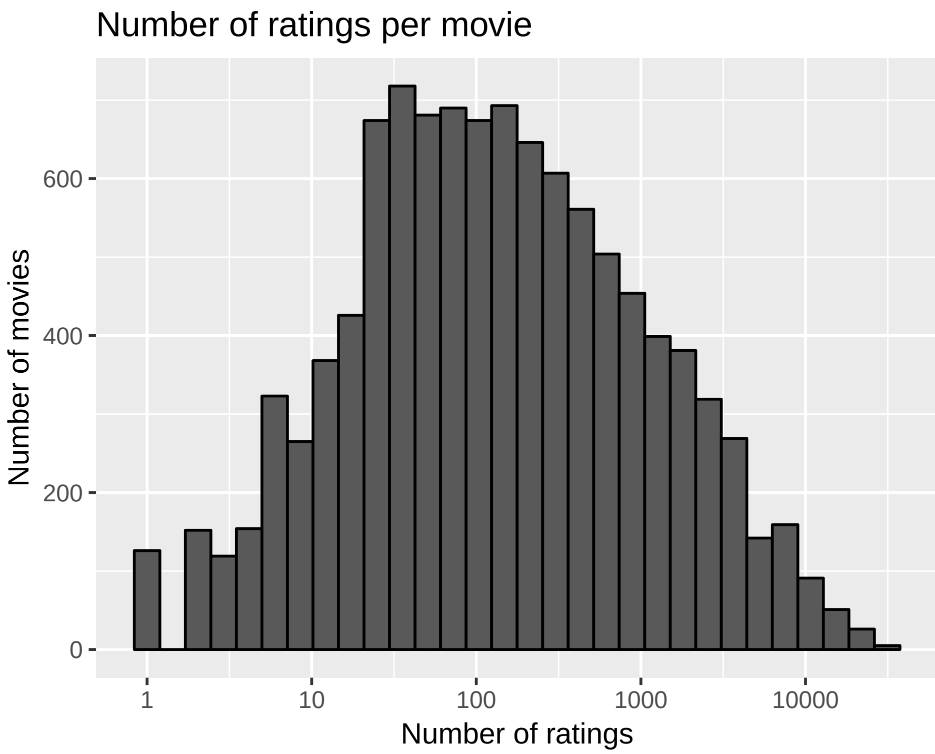
Users have a preference to rate movies rather higher than lower as shown by the distribution of ratings shown below. 4 is the most common rating, followed by 3 and 5. 0.5 is the least common rating. In general, half star ratings are less common than whole star ratings.



We can observe that some movies have been rated more often that others, while some have very few ratings and some have only one rating. In fact 125 movies have been rated only once. This is important to note as very low ratings might results in inaccurate estimates for our predictions.

For this reason, regularisation 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")
```



As 20 movies that were rated only once appear to be obscure, predictions of future ratings for them will be difficult.

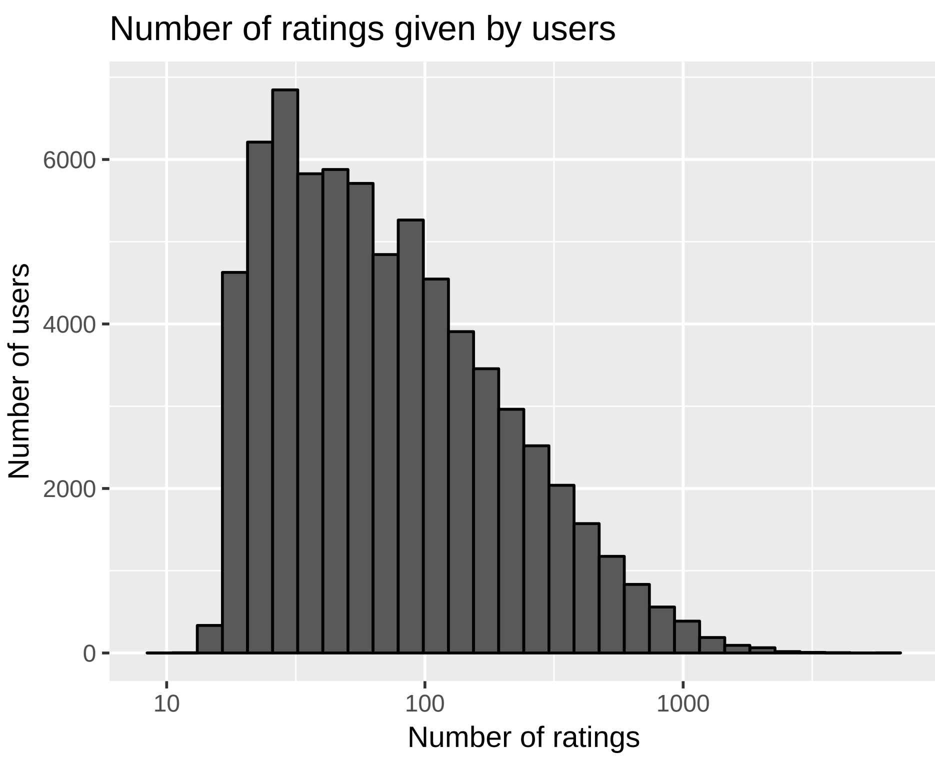
```
edx %>%
  group_by(movieId) %>%
  summarize(count = n()) %>%
  filter(count == 1) %>%
  left_join(edx, by = "movieId") %>%
  group_by(title) %>%
  summarize(rating = rating, n_rating = count) %>%
  slice(1:20) %>%
  knitr::kable()
```

title	rating	n_rating
1, 2, 3, Sun (Un, deuz, trois, soleil) (1993)	2.0	1
100 Feet (2008)	2.0	1
4 (2005)	2.5	1
Accused (Anklaget) (2005)	0.5	1
Ace of Hearts (2008)	2.0	1
Ace of Hearts, The (1921)	3.5	1
Adios, Sabata (Indio Black, sai che ti dico: Sei un gran figlio di…) (1971)	1.5	1
Africa addio (1966)	3.0	1
Aleksandra (2007)	3.0	1
Bad Blood (Mauvais sang) (1986)	4.5	1
Battle of Russia, The (Why We Fight, 5) (1943)	3.5	1
Bellissima (1951)	4.0	1
Big Fella (1937)	3.0	1
Black Tights (1-2-3-4 ou Les Collants noirs) (1960)	3.0	1
Blind Shaft (Mang jing) (2003)	2.5	1
Blue Light, The (Das Blaue Licht) (1932)	5.0	1

title	rating	n_rating
Borderline (1950)	3.0	1
Brothers of the Head (2005)	2.5	1
Chapayev (1934)	1.5	1
Cold Sweat (De la part des copains) (1970)	2.5	1

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.

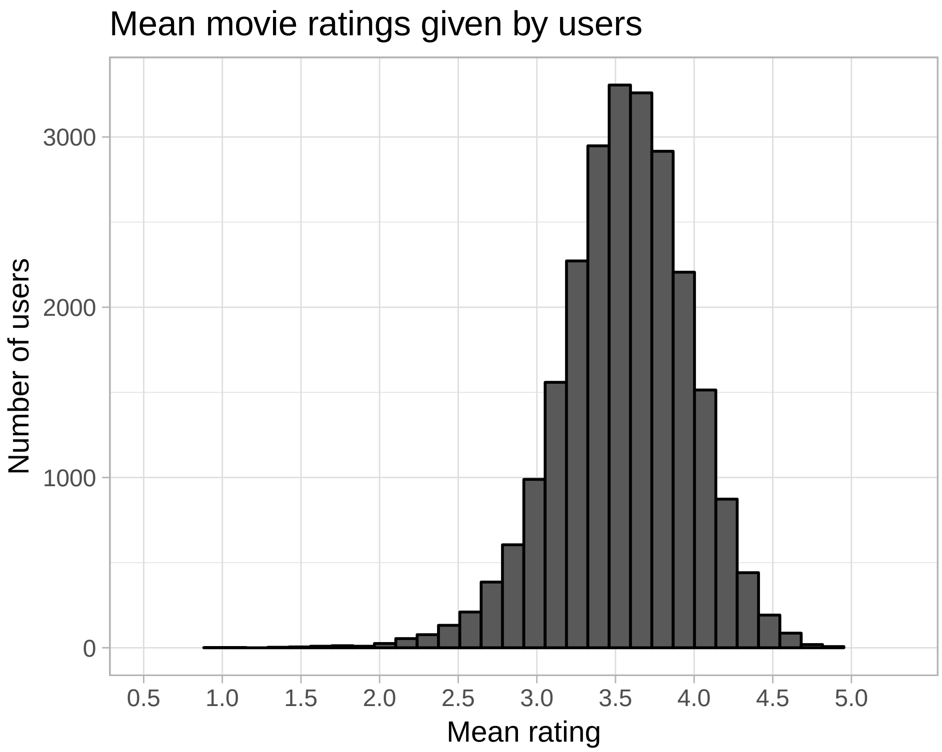
```
edx %>%
count(userId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "black") +
scale_x_log10() +
xlab("Number of ratings") +
ylab("Number of users") +
ggtitle("Number of ratings given by users")
```



Furthermore, users differ vastly in how critical they are with their ratings. Some users tend to give much lower ratings and some users tend to give higher ratings than average. The visualization below only includes 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()
```



Modelling Approach

We write now the loss-function, previously anticipated, that computes the RMSE:

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

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))
}
```

As said previously, the lower the better.

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 recommendation system by predicting the same rating for all movies regardless of the user(s) who gave it. A model based approach assumes the same rating for all movie with all differences explained by random variation :

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

with $\epsilon_{u,i}$ 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)
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
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.

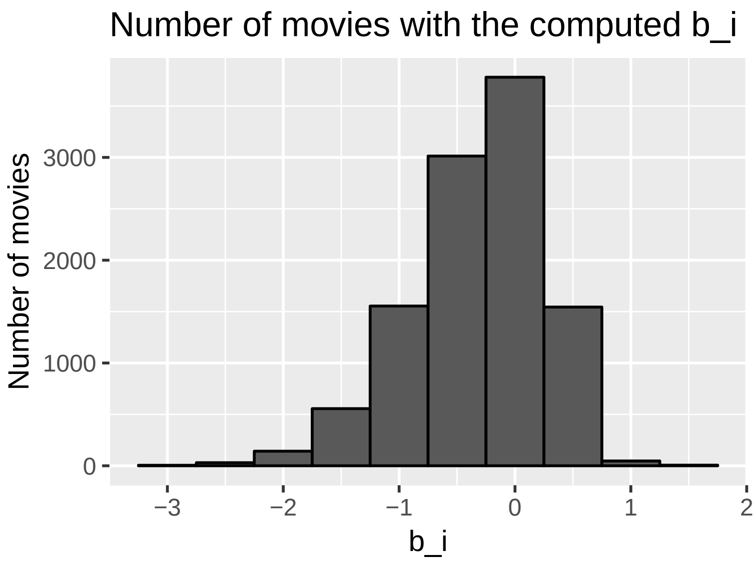
II. Movie effect model

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:

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

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

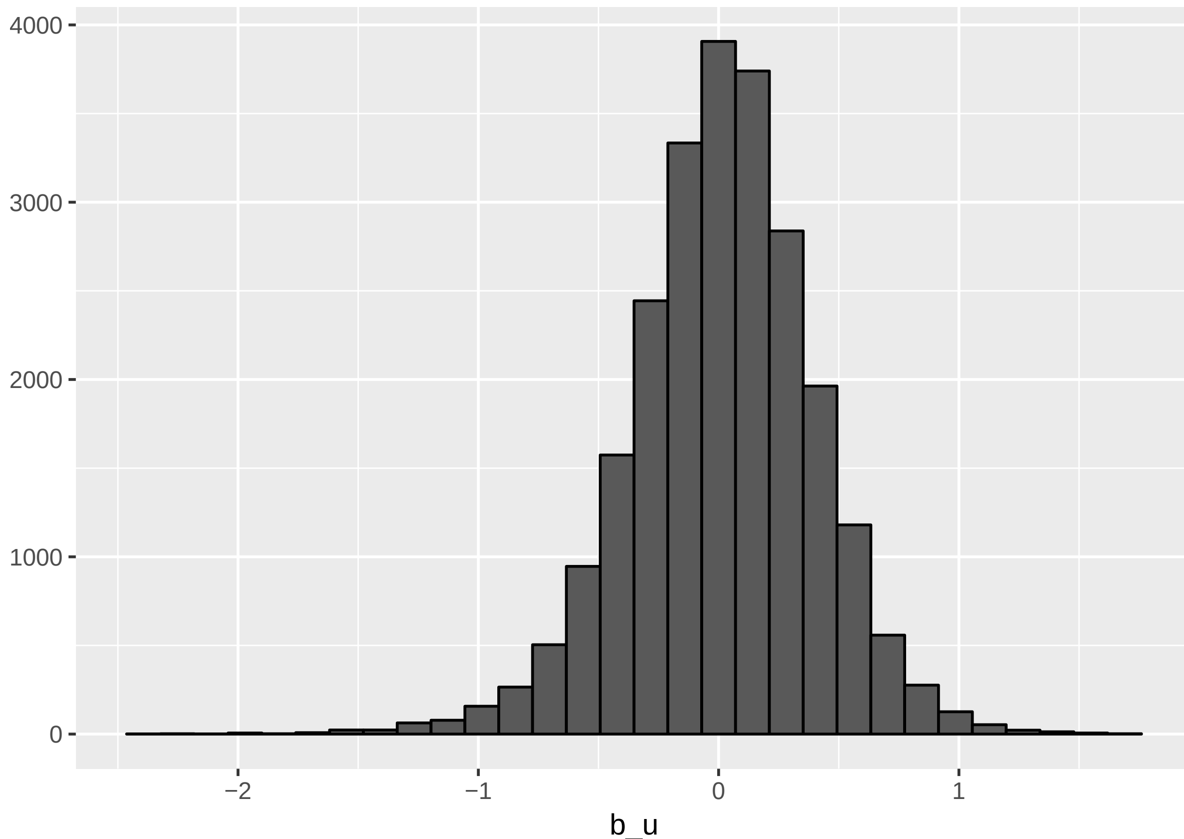
So we have predicted movie rating based on the fact that movies are rated differently by adding the computed b_i to μ. If an individual movie is on average rated worse that the average rating of all movies μ , we predict that it will rated lower that μ by b_i, the difference of the individual movie average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

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

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. In fact users 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 other love every movie. This implies that further improvement to our model may be:

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

where b_u is a user-specific effect. If a choosy 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 that the 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 supposed ???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.

Until now, we computed standard error and constructed confidence intervals to account for different levels of uncertainty. However, when making predictions, we need one number, one prediction, not an interval. For this we introduce the concept of regularization, that permits to penalize large estimates that come from small sample sizes. The general idea is to add a penalty for large values of b_i to the sum of squares equation that we minimize. So having many large b_i, make it harder to minimize. Regularization is a method used to reduce the effect of overfitting.

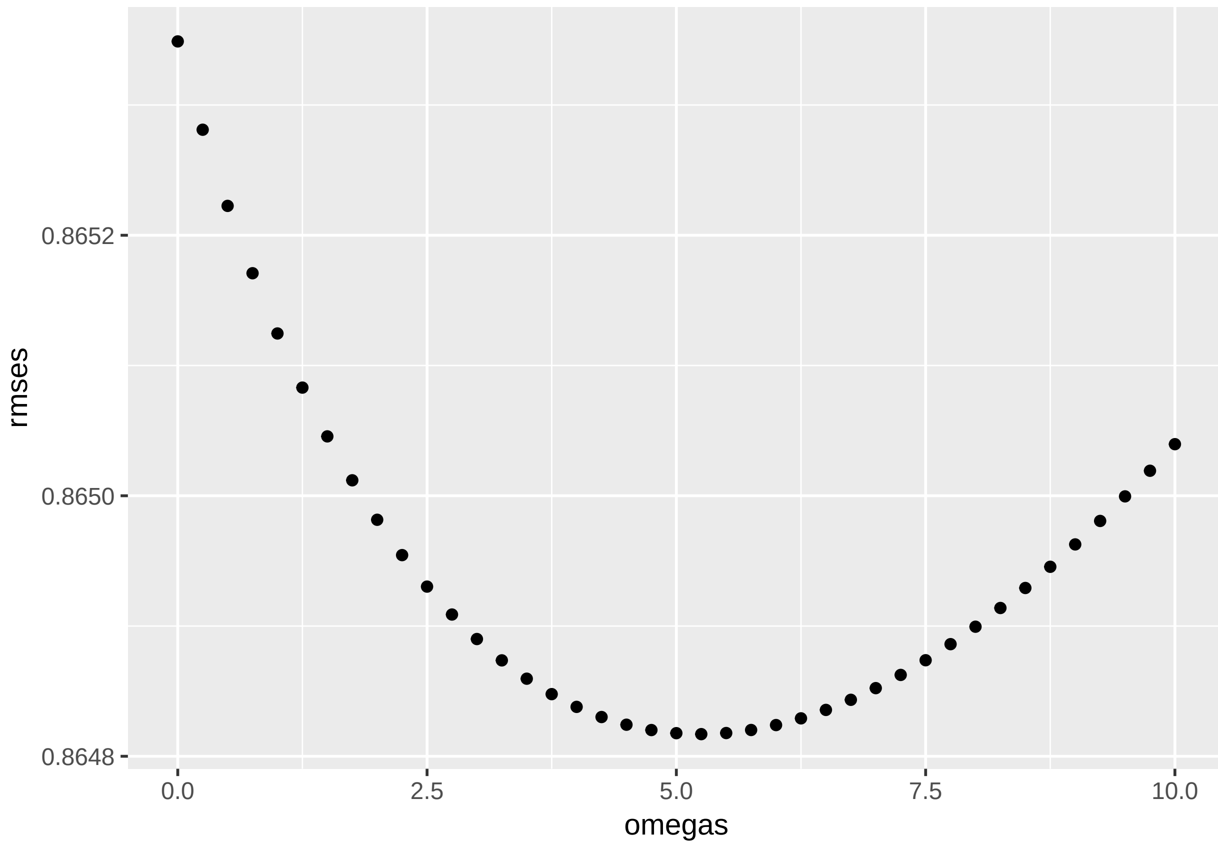
IV. Regularized movie and user effect model

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.

```
omegas <- seq(0, 10, 0.25)
rmses <- sapply(omegas, function(l){

  movavg <- 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 omegas to select the optimal omega

```
qplot(omegas, rmses)
```



For the final model, the optimal omega is:

```
  omega <- omegas[which.min(rmses)]
omega
```

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

For the final model, the optimal omega is: 5.25

The new results will be:

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularised 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
Regularised movie and user effect model	0.8648170

Results

The RMSE values of all the represented models:

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

We find the lowest value of RMSE to be at 0.8648170.

Conclusion

The final model for this project is:

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

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

We can affirm to have built a machine learning algorithm to predict movie ratings using the MovieLens dataset. 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. With this model, we have achieved our goal of creating an algorith with RMSE(0.8648170) lower than 0.8775. Improvements in the RMSE could be achieved by adding other effect (genre, year, age,..). Other different machine learning models could also improve the results further.