HarvardX PH125.9x Data Science Capstone Project: MovieLens

Francis Magombo

06/01/2021

Contents

1 Background **2**

2 Introduction **2**

2.1 The Objective of the project 2

2.2 Dataset for the analysis 2

3 Methods and Analysis **3**

3.1 Data Analysis 3

4 The Models **7**

4.1 Average movie rating model (Model 1) 7

4.2 Movie effect model (Model 2) 8

4.3 Movie and user effect model (Model 3) 9

5 Regularization **11**

5.1 Regularization of the movie and user effect model 12

5.2 Optimal lamba selection 12

6 Results **13**

7 Discussion and Conclusion **14**

8 Appendix- Environment **15**

1 Background

This project is related to the MovieLens Project of the Harvard: PH125.9x Data Science: Capstone course. The report will look at the objectives first and then review how the data set was prepared and made ready for the analysis.Thereafter the the report will focus on the data analysis to develop a the predictive machine learning algorithm leading to the final model. The final part will be on the explanation of the results before making concluding remarks.

2 Introduction

Recommendation systems use ratings that users have given to items to make specific recommendations. The extensive data sets collected through these systems can can be used to predict what rating a particular user will give to a specific item and those with higher ratings are then made part of a specific recommendation to the user. Recommendation systems are one of the most used models in machine learning algorithms and in this project we will use the data from MovieLens to make a recommendation though modeling.

2.1 The Objective of the project

The main objective of the project is to train a machine learning algorithm that predicts user ratings (from 0.5 to 5 stars) using the inputs of the provided subset of the data and to predict movie ratings using the validation subset of the data.

The value used to evaluate algorithm performance is the Root Mean Square Error(RMSE), one of the most used measure of the differences between the prediction by a model and the observed values. A lower RMSE is considered better. The RMSE is sensitive to outliers as larger errors have a disproportionately big effect since the RMSE effect of each error is proportional to the size of the squared error.

We will build and compare models using their resulting RMSE in order to assess find the best model among them. It is expected that we should at least have a RMSE lower than than 0.8775. RMSE will is defined as:

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

Here is the function that computes the RMSE for vectors of ratings and their corresponding predictors:

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

2.2 Dataset for the analysis

We will create a movie recommendation system using the 10M version of MovieLens dataset, collected by GroupLens Research.The dataset can be downloaded from the the following links: https://grouplens.org/datasets/movielens/10m/

http://files.grouplens.org/datasets/movielens/ml-10m.zip

Having downloaded and installed the packages the dataset will be split into a training set and a 10% validation (test) set.The code used to download the data is as outlined below:

```
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")
```

The MovieLens data set will be split into two subsets, the "edx", a training subset for the algorithm and "validation" a subset to test the movie ratings. This will help in making an accurate prediction for the movie ratings. The "validation subset will be 10% of the MovieLens data.

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

3 Methods and Analysis

3.1 Data Analysis

First we conduct an exploratory data analysis to get a picture of what we have in the data set in terms of the variables and the rows.We see that the the data set has six variables: "userID", "movieID", "rating", "timestamp", "title", and "genres". The first 6 rows of the "edx" subset are shown below. Each row represent a single rating of a user for a single movie.

Here is the list of all variables and

x
userId
movieId
rating
timestamp
title
genres

The first 6 rows of the "edx" dataset are shown below:

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 1 Comedy|Romance 2 Action|Crime|Thriller 3 Action|Drama|Sci-Fi|Thriller 4 Action|Adventure|Sci-Fi 5 Action|Adventure|Drama|Sci-Fi 6 Children|Comedy|Fantasy

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

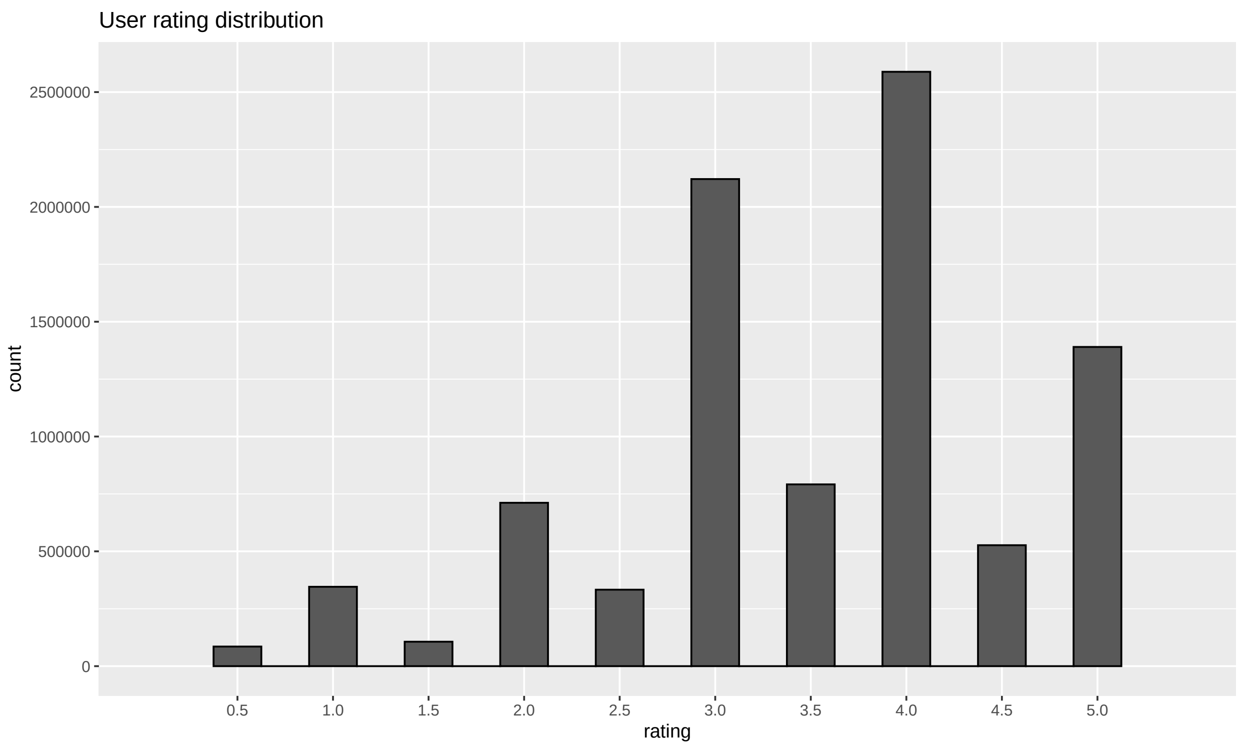
There are no missing values in the subset as evidenced through the summary below:

	x
userId	0
movieId	0
rating	0
timestamp	0
title	0
genres	0

We see that there are 69,878 unique users and about 10,677 different movies in the edx data set.

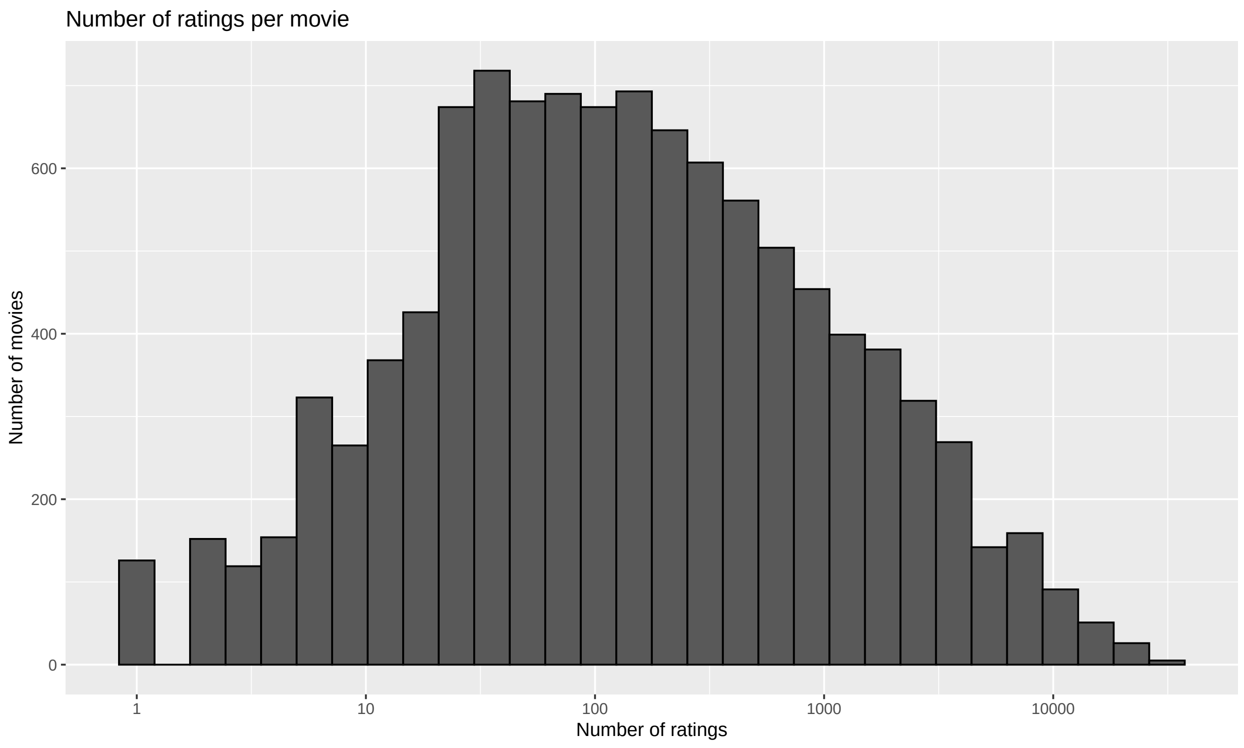
n_users	n_movies
69878	10677

We create a graph to show the rating distribution for different users. From the graph we can see that 4 is the median rating. The rating of 3 and 5 follow with 0.5 and 1.5 being the least common rating among the users.



Regularization will need to be done due to some of the issues in the data as outlined in the paragraphs below.Regularization is used to reduce the error by fitting a function appropriately on the given training set and avoid overfitting. Regularization tunes the function by adding an additional penalty term in the error function to limit and control extreme values in the analysis.

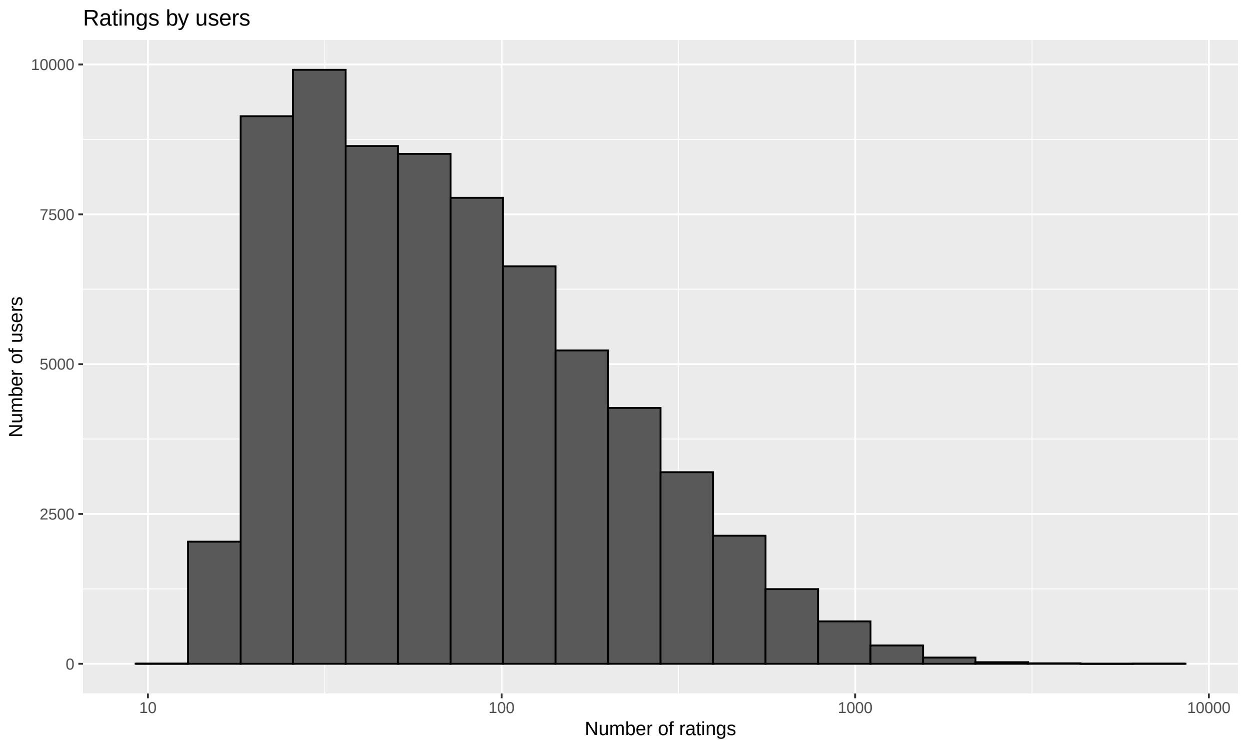
In terms of the frequency of movie ratings we find that there is a wide variation in how often the movies are rated. Some movies are rated more often while others are rated only once. A total of over 100 movies fall into this group. These would be taken as an outliers and may cause inaccurate prediction results for our models. In order to overcome this problem it will be important to apply a regularization and a penalty term to the models.



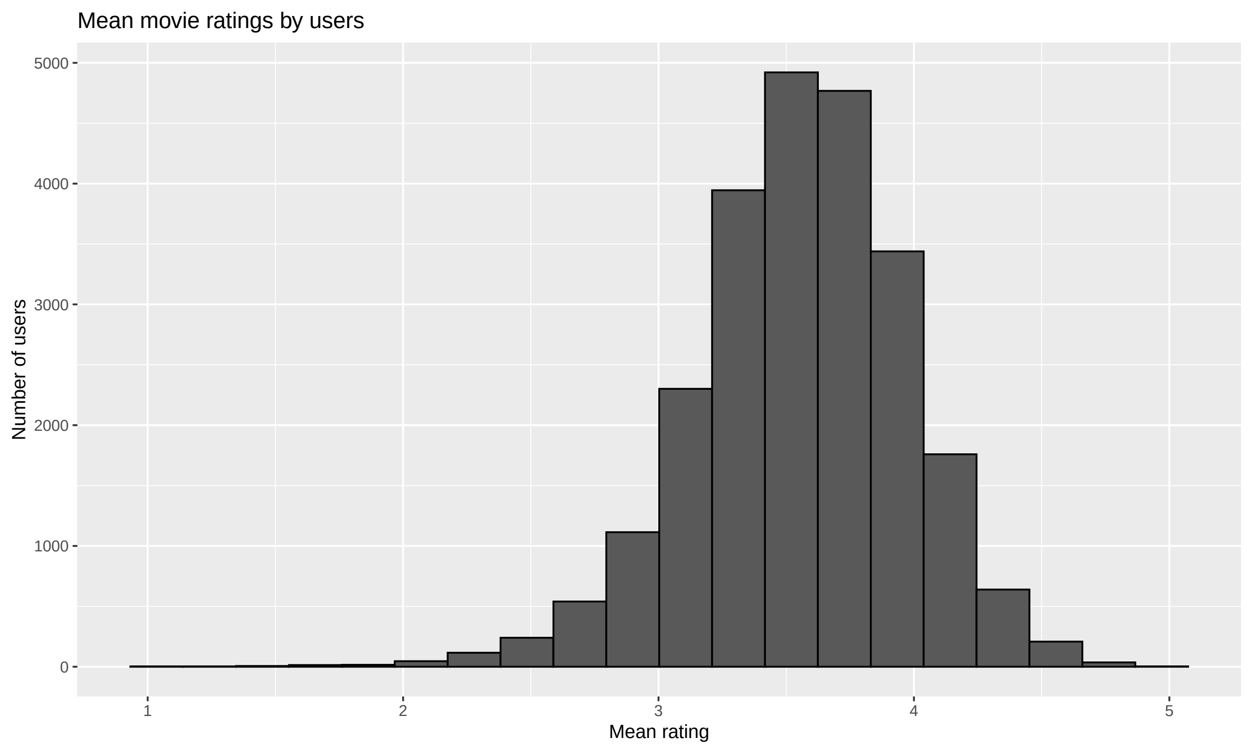
Some of the movies that were rated only once could also affect the prediction giving an inaccurate prediction. We isolate 20 obscure movies that were rated only once in the table below.

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
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

Most users have rated between 30 and 100 movies. This too means a user penalty term needs to be included the models. the plot below shows the number of ratings given by users.



In order to get a more reliable situation for the users' rating we will review the ratings of the users who have given 100 or more reviews. This will help in evening out the variation in individual user bias as some users tend to give much lower star ratings and others higher star ratings than average. The graph below shows the mean movie ratings given by users.



4 The Models

As stated earlier on, we will use the Residual Mean Squared Error(RMSE) as a measure of accuracy for our prediction. The RMSE can be interpreted in a similar way as a standard deviation since it is the typical error we make when making predictions like in the movie rating. We will aim for the lowest RMSE. The RMSE for vectors of ratings and their corresponding predictions is computed as follows:

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

4.1 Average movie rating model (Model 1)

The first model will be a simplest recommendation systems that will let us predict the same rating for all movies regardless of user.This will be done by calculating the dataset's mean rating. The expected rating of the underlying data set is between 3 and 4. This model 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.The estimate that minimizes the RMSE is the least square estimate of $Y_{u,i}$, in this case, is the average of all the ratings which in our case will be computed using the ode below:

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

```
[1] 3.512465
```

The mean rating, μ, in our case is 3.512465.If we predict all unknown ratings with μ or mu, we obtain the first naive RMSE. We then present the results table with the first RMSE:

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

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

method	RMSE
Average movie rating model	1.061202

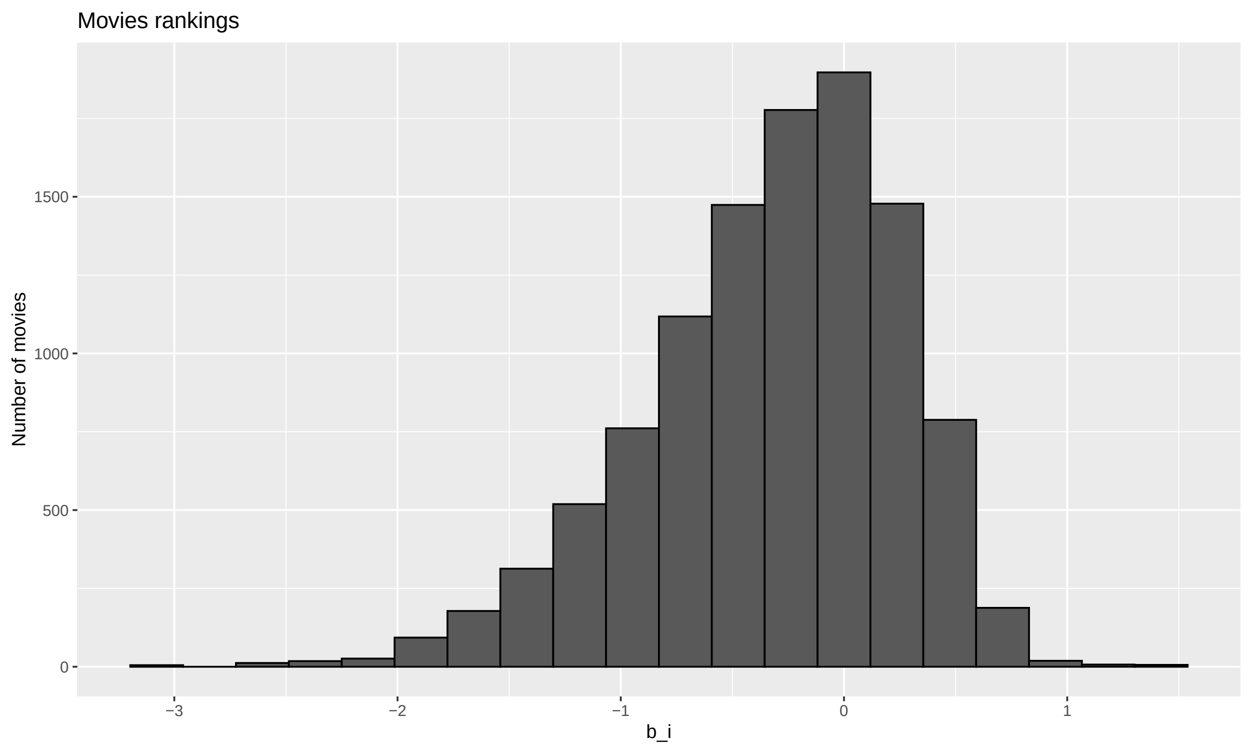
The naive RMSE is then 1.061202. We will need to compare this result of this naive approach with those from other different approaches of modelling our prediction. The subsequent models will need to be better than the naive model. So we will need to take into consideration other factors and the findings from the exploratory data analysis. We start with the movie effect as we are now aware that some movies are just generally rated higher than others.

4.2 Movie effect model (Model 2)

Higher ratings are mostly linked to popular movies among users and the opposite is true for unpopular movies. We improve the above model by adding the term b_{i} to represent average ranking for movie i to make the formula as shown below.We then calculate the estimated deviation of each movies' mean rating from the total mean of all movies μ.

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

This is a simple model taking into account the movie effect b_{i}. We will subtract the rating minus the mean for each rating the movie received and plot number of movies with the computed b_{i}. The histogram is left skewed, implying that more movies have negative effects.



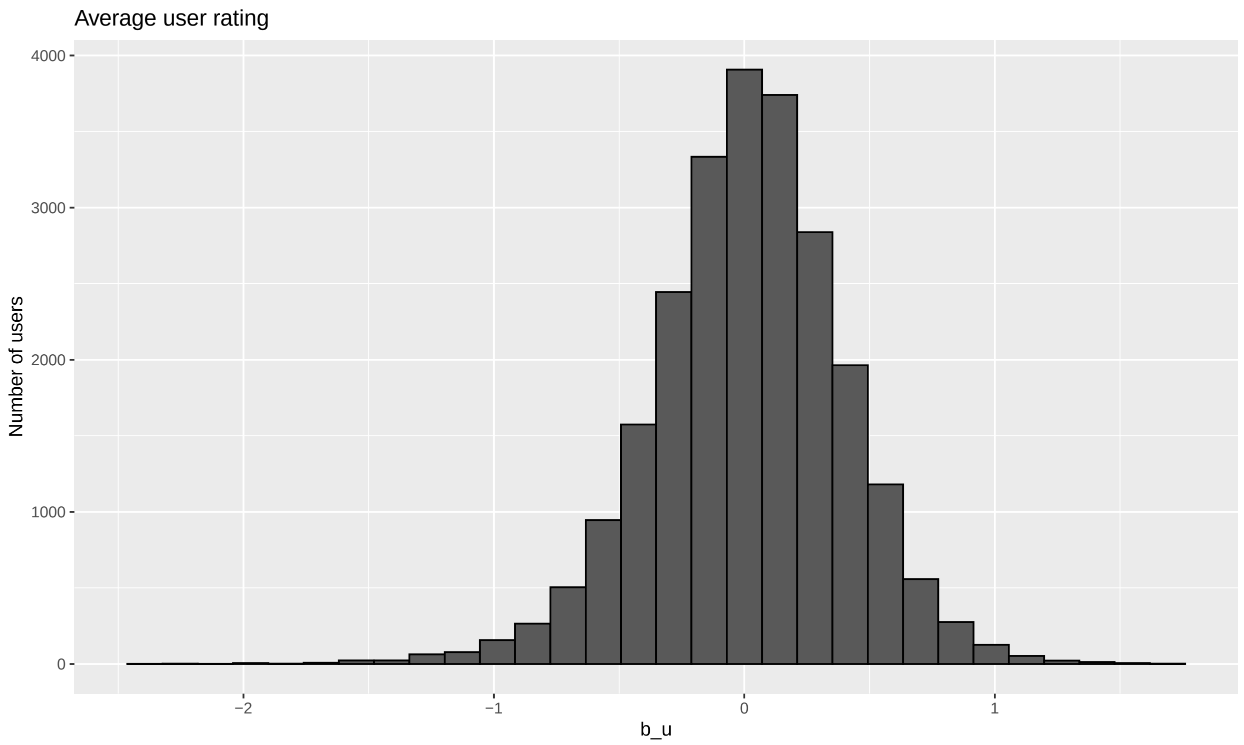
```
predicted_ratings <- mu +  validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results2 <- bind_rows(rmse_results,
                          tibble(method="Movie effect model",
                                     RMSE = model_2_rmse ))

rmse_results2 %>% knitr::kable()
```

method	RMSE
Average movie rating model	1.0612018
Movie effect model	0.9439087

This prediction improves upon the results of Model 1, giving us a lower RMSE of 0.9439087. If an individual movie is on average rated worse than the average rating of all movies μ , we predict that it will rated lower than μ by b_i, the difference of the individual movie average from the total average and vice versa. However, we may still improve upon this model by considering individual user effect in addition.

4.3 Movie and user effect model (Model 3)

We compute the average rating for user μ, focusing only on the users who have rated over 100 movies, said penalty term user effect. This is because there is usually also a bias on how individual users make their ratings and this affects the ratings positively or negatively.We will use the term b_{u} to represent average rating for the user u which is plotted below.



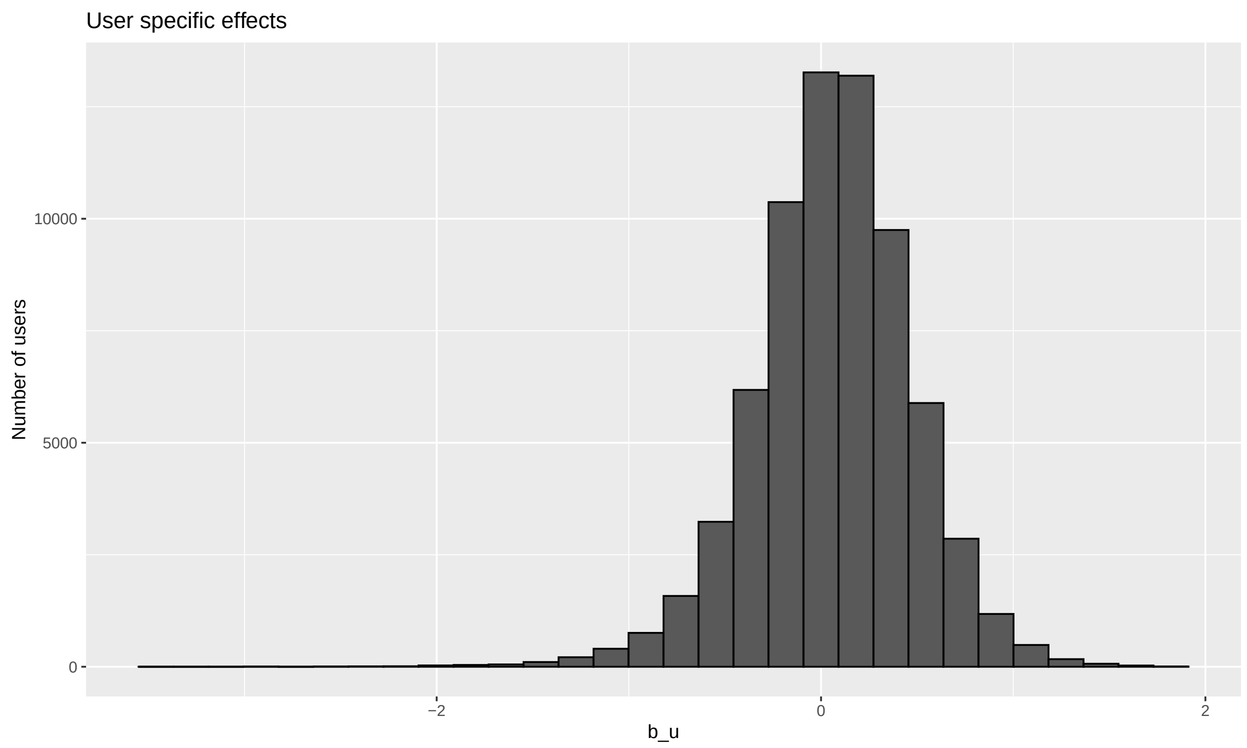
In order to take care of the variability across users we can improve the model as by adding b_u where the symbol $b_{u} is the user-specific effect and write the model as follows:

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

A user may unexpectedly give a negative b_u rates to a great movie that otherwise requires positive b_i). The effects will counter each other and we may be able to correctly predict that this user gave this great movie a 3 rather than a 5. Our calculation will be to approximate μ and b_i, and make an estimate of b_u, as the average of

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

We thus include the user specific effects in the model as follows:



The predictors and the movie and user effects model RMSE will then be computed as follows to see if there is the expected further improvement on the accuracy. The model gives an RMSE of 0.8653488.

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

```r
model_3_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results3 <- bind_rows(rmse_results2,
                          tibble(method="Movie and user effect model",
                                 RMSE = model_3_rmse))
rmse_results3 %>% knitr::kable()
```

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

5 Regularization

Our rating predictions further reduced the RMSE. But our first model using only movies effects included movies that were rated by one or very few users and the movies themselves may have been obscure ones. This creates a biased rating for the movies that were considered to be the best or even the worst since few users influencing the ratings we have more uncertainty. Therefore larger estimates of b_i, negative or positive, are more likely.These large errors can increase our RMSE.

Regularization penalizes large estimates that come from small sample sizes and also reduces the effect of overfitting models. The principle is to constrain the total variability of the effect sizes and 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. The general idea of penalized regression is to control the total variability of the movie effects.

5.1 Regularization of the 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. Therefore, we will use regularization to penalize these effects. However, first we must find the value of lambda (λ), the tuning parameter, which will be the best to minimize the RMSE. The Lambda will shrink the b_i and b_u to take care of the very few ratings that are influencing the inaccurate predictions.When our sample size is very large, a case which will give us a stable estimate, then the penalty, λ, is ignored but when the number of ratings is small, then the estimate is shrunken towards 0. We will use cross-validation to choose the most appropriate lambda.

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

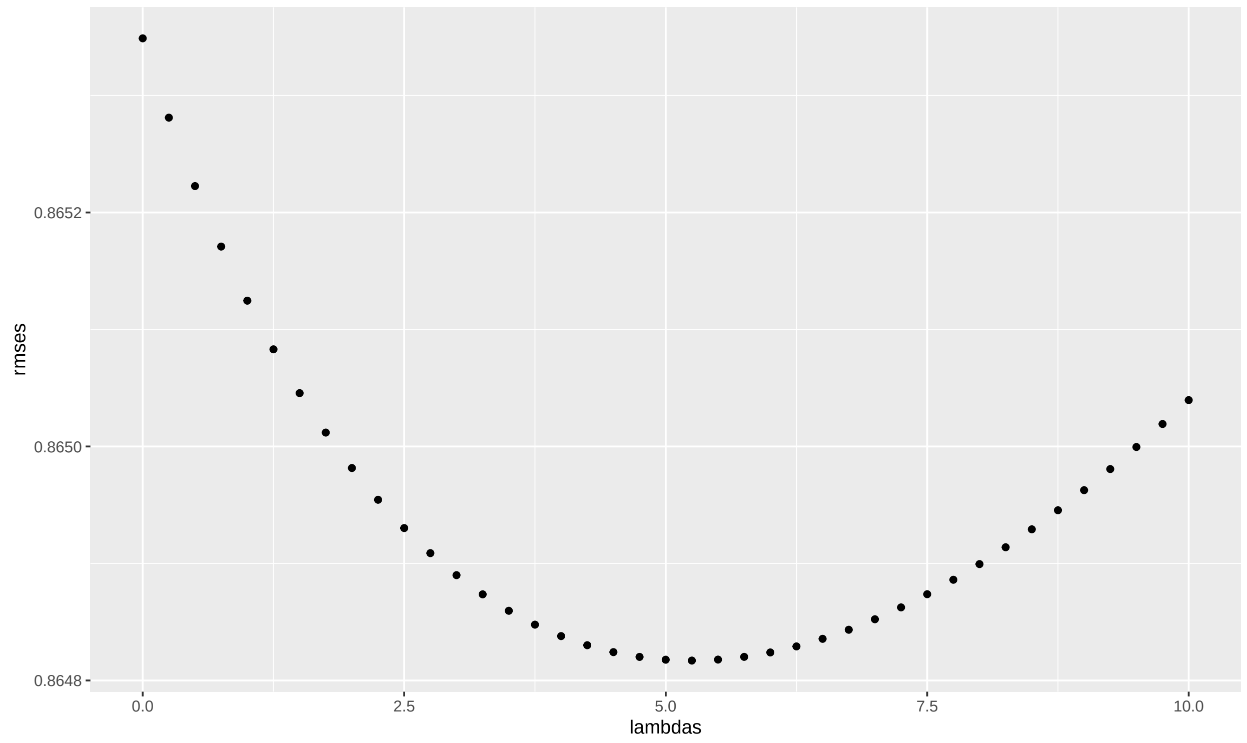
For each lambda we will calculate b__i & b__u, followed by rating prediction and testing

```r
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

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

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    dplyr::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))
})
```

5.2 Optimal lamba selection

We then plot RMSE vs lambdas to select the optimal lambda and the result is displayed below.

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



As we can see from the results of the calculation of the minimum lambda the lambda that minimises the RMSE is 5.25.

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

[1] 5.25

From this the regularised RMSE results for the movie and user effect model will now be calculated. The result is 0.8648170.

```
rmse_results4 <- bind_rows(rmse_results3,
                           tibble(method="Regularized movie and user effect model",
                                  RMSE = min(rmses)))

rmse_results4 %>% 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

6 Results

The RMSE values of all the represented models are presented in the regularised model. The lowest value of RMSE that is 0.8648170 and the regularised model is the one that is the most effective in making the prediction and the one that we will use for the movie recommendation.

7 Discussion and Conclusion

The following is the final model we can propose for our prediction model project.

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

The regularized model including the effect of user has the lower RMSE value and is hence the best model to use for the present project. This model has the lowest RMSE value (0.8648170) lower than the initial evaluation criteria (0.8775) given by the goal of the present project.

Improvements in the RMSE can be achieved by adding other effects including genre, year, age among others.

8 Appendix- Environment

```
[1] “Operating System:” _
platform x86_64-w64-mingw32
arch x86_64
os mingw32
system x86_64, mingw32
status
major 4
minor 0.4
year 2021
month 02
day 15
svn rev 80002
language R
version.string R version 4.0.4 (2021-02-15) nickname Lost Library Book
```