MovieLens (edx Harvard Data Science Capstone Project)

Osama AlJariri

05/01/2019

Introduction

In this project we will build recommendation system for rating movies, we will use ratings given by users for movies, and recommend movies to users as per their ratings to other movies; it will predict the movies rating for a specific user, and movies with high rating will be recommended to the user. Our goal in this project is to minimise the RMSE value to be less than 0.8649.

In this project we will use movielens dataset, we will use the 10M version of the datset. each row in the dataset represents one rating given by one user to one movie, and has the columns 'Userid', "Movieid", "Title", "Rating", "TimeStamp" and "Genres". The relation between users and movies is many to many, implies that one user can rate many movies and one movie can be rated by many users.

We divided the dataset to training and validation data, and start to evaluate the RMSE using the below models
- Using Movies effect model
- Using Movies and users effect model
- Using Regularized movie effect model
- Using Regularized movie effect and user effect model

edx Dataset

Below is the first 5 rows of the dataset,

```
##   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
```

The table below shows the ditinct number of movies and users that included in the dataset,

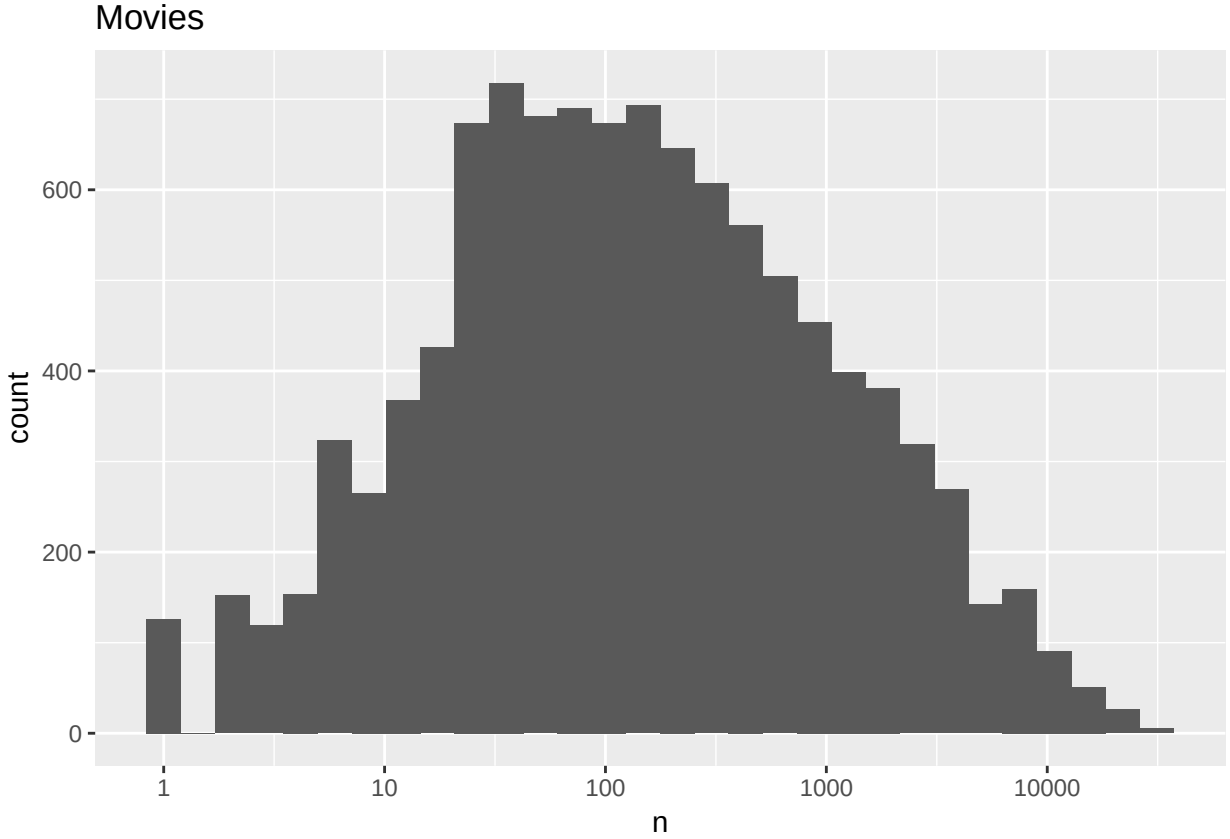
```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector

## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

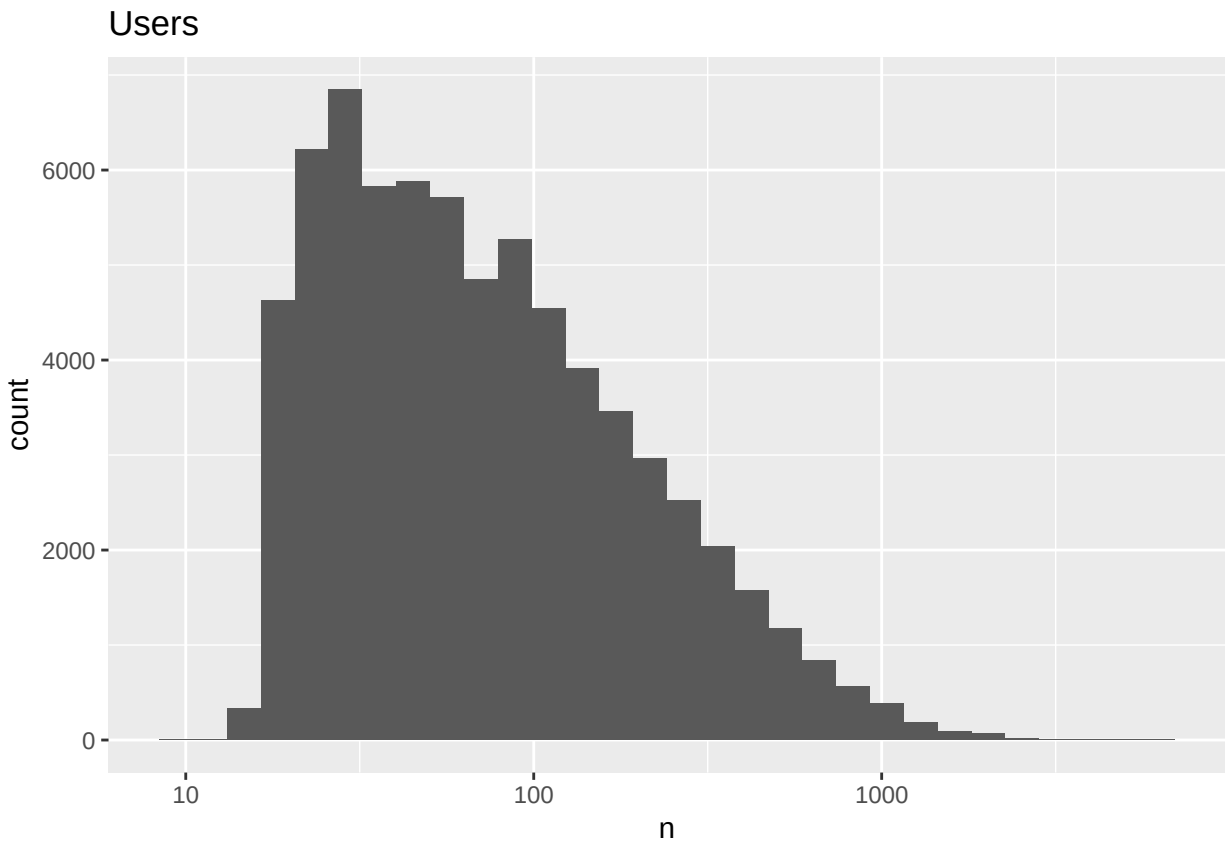
Data	Count
Movies	10677
Users	69878

Data Properties

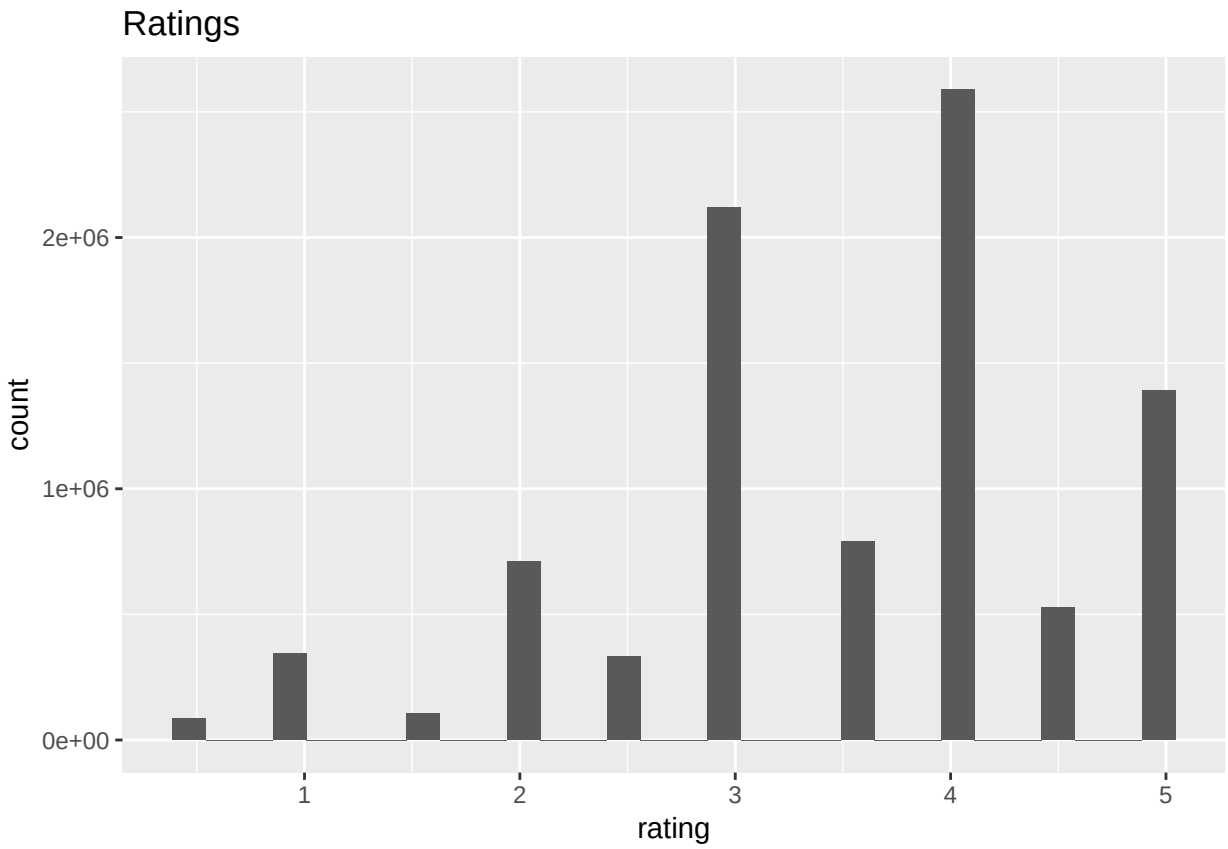
In this section we will figure out some properties for the dataset that we have, the first thing we note about the movies ratings is that number of ratings are varies among movies as some movies are very popular and watched by millions and some movies are just watched by a few, as shown below



Also we noticed that number of ratings done by users are varies as some users are more active than others, as shown below



The last thing we noticed is that whole star rating (ex: 3, 4, 5) was given more than half star rating(ex: 1.5, 3.5, 4.5), as shown below



Analysis

We will use the movielens dataset to do our analysis, we create training dataset (edx) and validation data set (validation), and we will build different models to calculate the resual mean squared error (RMSE), which is the error when predicting a movie rating, its value indicates the number of stars that our prediction is away from the correct rating. the RMSE function is shown below

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

We will compare the RMSE values for different models and see if we can achieve our goal of having RMSE value to be less than 0.8649.

Lets start with the models gradually from simplest one:

Average Rating Model

In this model we will calculate the average ratings for all movies done by all users and compare each rating to that average to calculate the RMSE. The estimate that minimizes the residual mean squared error is the least square estimate of mu, in this case its the average of all the ratings, we can find the average as below

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

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

to calculate the RMSE we use the test data as below

```r
naive_RMSE <- RMSE(validation$rating, mu_hat)
naive_RMSE
```

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

Note that if we used any value other than the average we will get higher RMSE, because we know that the average minimizes the RMSE, check below using value of 2.5 to find the RMSE

```r
testRMSE <- RMSE(validation$rating, 2.5)
testRMSE
```

```
## [1] 1.46641
```

We can clearly notice the high value of the RMSE using the average rating which is 1.06, now we wil try other models to try achieve RMSE of less than 0.8649.

To have a record for all our models results, we create a table that has all our results

```
## [1] "Using the \"Average\" effect model we have achived a RMSE of value 1.06120181029262"
```

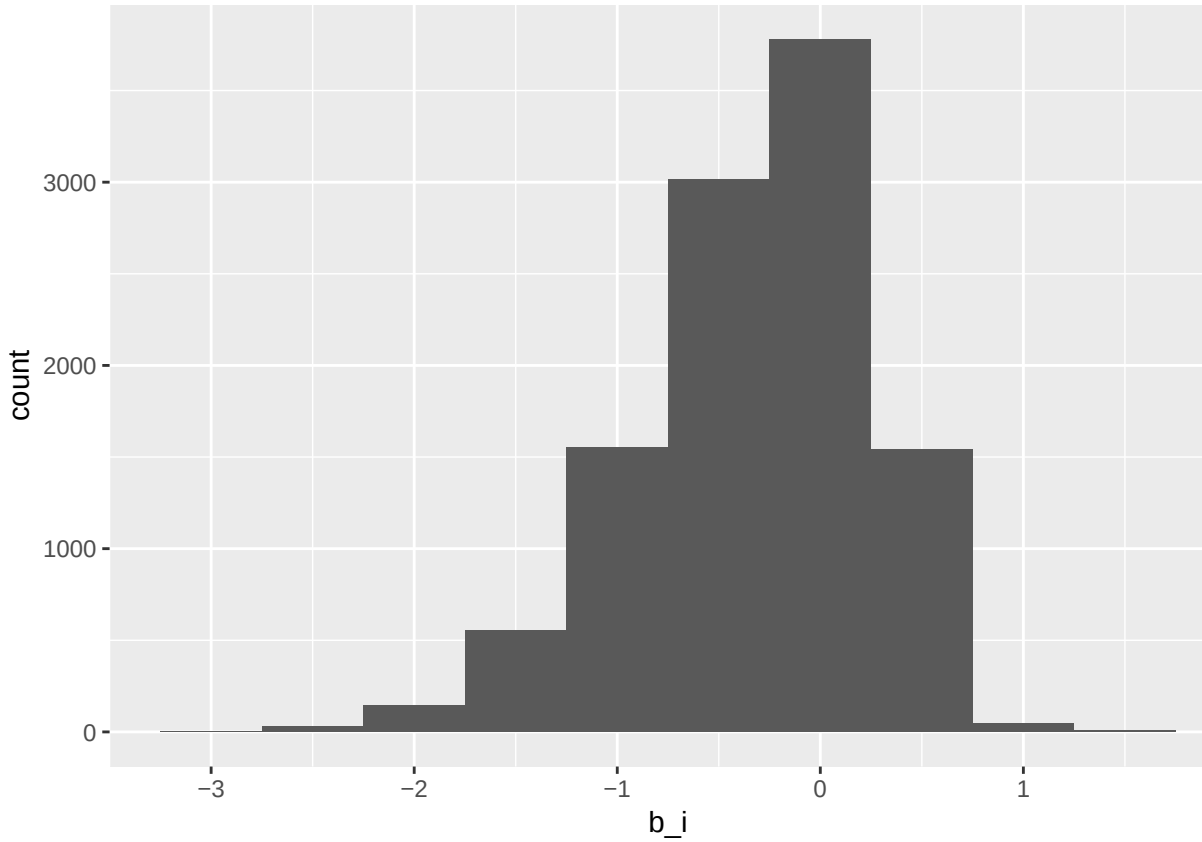
Average Rating with Movie effect Model

we showed that each movie has been rated differently than others (Plot in Data Properties section)

That implies that average rating for specific movie, plays an important factor of building our module, as if we included to our model to calculate the RMSE we will get the following

```r
mu <- mean(edx$rating)
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarise(b_i= mean(rating-mu))
```

Lets plot the histogram values of the movie rating biases (b_i), its almost binormal distribution, most of the movies has the bias of 0, some of them has a bias of 1.5(which is the maximum value that when added to average will have rating of 5), and some of them has the bias of -3.5(minimum bias that when added to the average



will give rating of 0)

We now calculated the predicted ratings by adding the mean value to b_i (which is the bias for the movie effect) for each movie. Then we calculate the RMSE between the predicted and actual ratings.

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

```
movie_effect_RMSE <- RMSE(predicted_ratings, validation$rating)
```

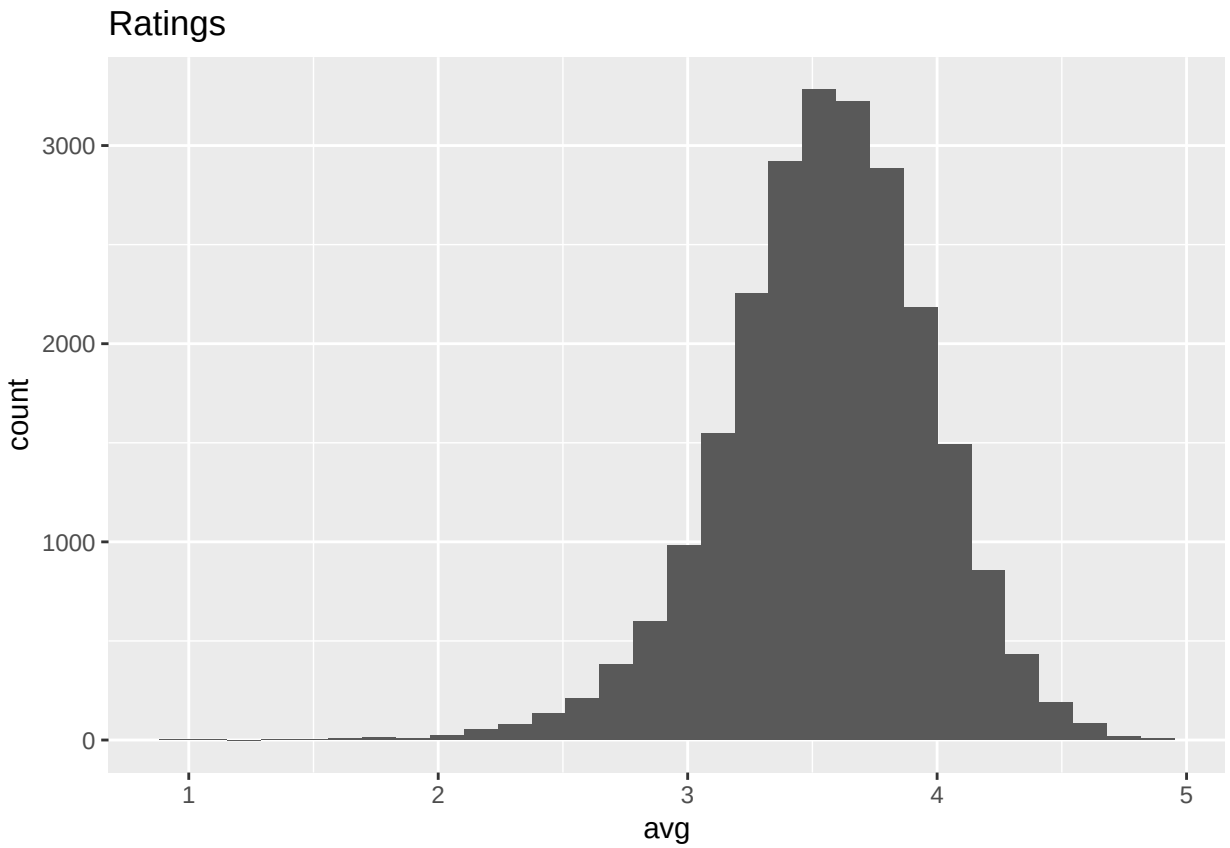
```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

```
## [1] "Using the \"movie\" effect model we have achived a RMSE of value 0.943908662806309"
```

we can clearly notice that the RMSE value is getting improved.

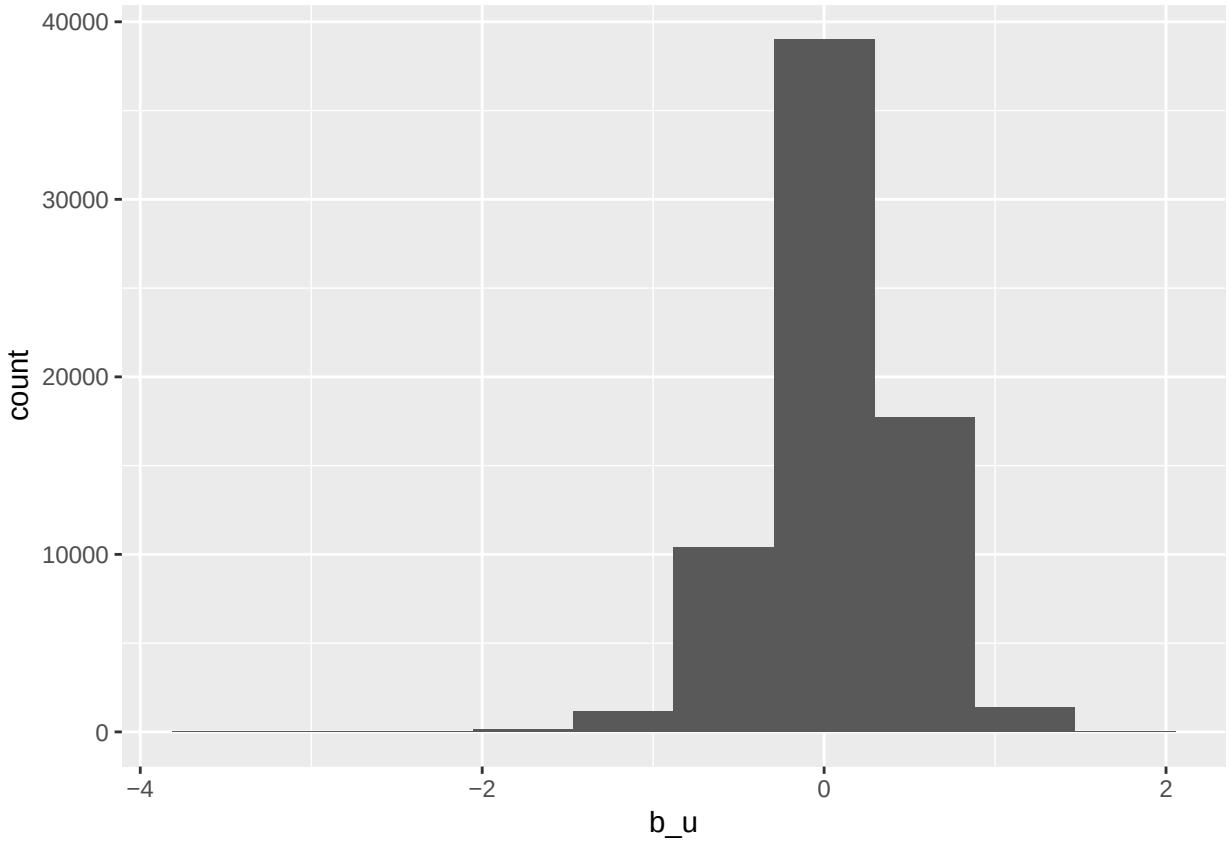
Average Rating with Movie and User effect Model

The below plot shows the average rating for users who have more than 100 ratings, and we can clearly not the substantial difference in users preferences.



We noticed from the user rating plot, that Number of ratings done by users are varies as users have different movies preferences.so we can add another factor beside the movie bias, which is the user bias, as each movie rating affected by the difference between movie average rating the mean (u_i), and also by the difference between the users ratings and the mean which is the user bias b_i.

Lets calculate the user effect biases, and plot their averages.



Now, lets calculate the RMSE based on the movie and user effects, and use the biases we calculated before, we notice from the table below that we did a good improvement in the RMSE value.

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

```
movie_user_effect_RMSE <- RMSE(predicted_ratings, validation$rating)
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

```
## [1] "Using the \"movie + User\" effect model we have achived a RMSE of value 0.865348824577316"
```

Regularization

We have achieved good results with the movie based effective modules, lets now focus on how can we improve those modules, lets view the movies that have the highest residuals (difference between rating and expected rating)

title	residual
From Justin to Kelly (2003)	4.097990
From Justin to Kelly (2003)	4.097990
PokÃ©mon Heroes (2003)	3.970803
PokÃ©mon Heroes (2003)	3.970803
Shawshank Redemption, The (1994)	-3.955131
Shawshank Redemption, The (1994)	-3.955131
Shawshank Redemption, The (1994)	-3.955131
Shawshank Redemption, The (1994)	-3.955131
Shawshank Redemption, The (1994)	-3.955131
Shawshank Redemption, The (1994)	-3.955131

Now lets view the best ten movies according to the movie bias values, we notice that most of these movies are not well known

title	b_i
Hellhounds on My Trail (1999)	1.487535
Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	1.487535
Shadows of Forgotten Ancestors (1964)	1.487535
Fighting Elegy (Kenka erejii) (1966)	1.487535
Sun Alley (Sonnenallee) (1999)	1.487535
Blue Light, The (Das Blaue Licht) (1932)	1.487535
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.237535
Human Condition II, The (Ningen no joken II) (1959)	1.237535
Human Condition III, The (Ningen no joken III) (1961)	1.237535
Constantine's Sword (2007)	1.237535

Now lets view the worst ten movies according to the movie bias values, we notice that most of these movies are also not well known

title	b_i
Besotted (2001)	-3.012465
Hi-Line, The (1999)	-3.012465
Accused (Anklaget) (2005)	-3.012465
Confessions of a Superhero (2007)	-3.012465
War of the Worlds 2: The Next Wave (2008)	-3.012465
SuperBabies: Baby Geniuses 2 (2004)	-2.717822
Hip Hop Witch, Da (2000)	-2.691037
Disaster Movie (2008)	-2.653090
From Justin to Kelly (2003)	-2.610455
Criminals (1996)	-2.512465

We can conclude from the above plots that based on using movie effect only , the best and worst movies ar odd and not well known movies, and we can prove that by the below plots on the count of votes those movies have, so this is one factor that make the movie effect model have little impact on the RMSE, as these odd movies has little number of rates and caused the deviation of the calculations.

Count of best Movies according to b_hat

```
## Joining, by = "movieId"
```

title	b_i	n
Hellhounds on My Trail (1999)	1.487535	1
Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	1.487535	2
Shadows of Forgotten Ancestors (1964)	1.487535	1
Fighting Elegy (Kenka erejii) (1966)	1.487535	1
Sun Alley (Sonnenallee) (1999)	1.487535	1
Blue Light, The (Das Blaue Licht) (1932)	1.487535	1
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.237535	4
Human Condition II, The (Ningen no joken II) (1959)	1.237535	4
Human Condition III, The (Ningen no joken III) (1961)	1.237535	4
Constantine's Sword (2007)	1.237535	2

Count of worst Movies according to b_hat

```
## Joining, by = "movieId"
```

title	b_i	n
Besotted (2001)	-3.012465	2
Hi-Line, The (1999)	-3.012465	1
Accused (Anklaget) (2005)	-3.012465	1
Confessions of a Superhero (2007)	-3.012465	1
War of the Worlds 2: The Next Wave (2008)	-3.012465	2
SuperBabies: Baby Geniuses 2 (2004)	-2.717822	56
Hip Hop Witch, Da (2000)	-2.691037	14
Disaster Movie (2008)	-2.653090	32
From Justin to Kelly (2003)	-2.610455	199
Criminals (1996)	-2.512465	2

What we will do to eliminate the distracting data, is to add a penalty for the large values of movie bias, so when the number of observations for the same movie are large the the penalty term (lamda) goes to zero.

Regulization with movie effect

We will divide the edx data into train and test data, and at the end we will use the validation data.

```
# preparing the training data and test data the will use cross validation to determine the best lambda.

set.seed(1, sample.kind="Rounding")
```

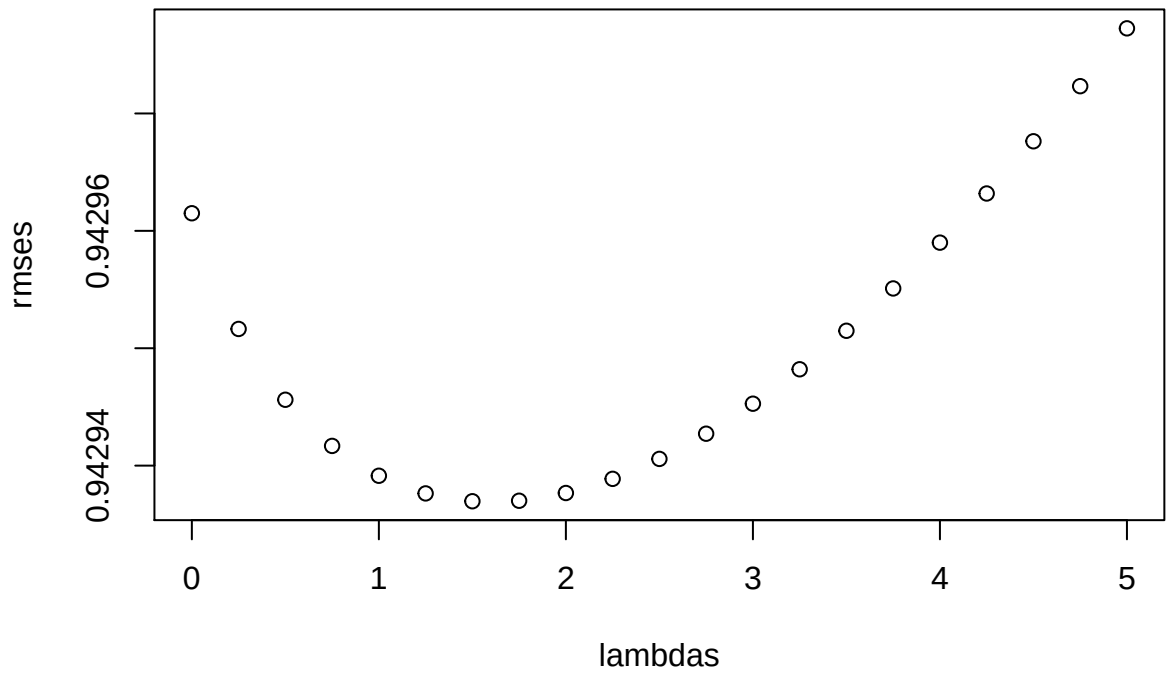
```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
test_index_Reg <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
edx_Reg <- edx[-test_index_Reg,]
temp_Reg <- edx[test_index_Reg,]


test_Reg <- temp_Reg %>%
```

```
  semi_join(edx_Reg, by = "movieId")


removed_Reg <- anti_join(temp_Reg, test_Reg)
```

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

```
edx_reg <- rbind(edx_Reg, removed_Reg)

rm(test_index_Reg, temp_Reg, removed_Reg)


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

lambdas <- seq(0, 5, 0.25)

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


  mu <- mean(edx_Reg$rating)


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


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

  return(RMSE(predicted_ratings, test_Reg$rating))
})

plot(lambdas, rmses)
```




```
lambda <- lambdas[which.min(rmses)]
paste('Optimal RMSE of',min(rmses),'is achieved with Lambda',lambda)
```

```
## [1] "Optimal RMSE of 0.942936965913324 is achieved with Lambda 1.5"
```

```
# Now we will use the lambda to calculate the RMSE using the **Validation** data

l <- lambda


  mu <- mean(edx$rating)

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

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


finalRMSE_movie = RMSE(predicted_ratings, validation$rating)
```

We can check the rate for the top 10 movies using the regulized movie effect module, we notice that the list contains “Shawshank Redemption”, “Godfather, The”, “Usual Suspects, The”, “Schindler’s List”, “Casablanca” which makes more sense.

```
## Joining, by = "movieId"
```

title	b_i	n
More (1998)	0.9897345	7
Shawshank Redemption, The (1994)	0.9426155	28015
Godfather, The (1972)	0.9028245	17747
Who’s Singin’ Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	0.9000253	4
Human Condition II, The (Ningen no joken II) (1959)	0.9000253	4
Human Condition III, The (Ningen no joken III) (1961)	0.9000253	4
Usual Suspects, The (1995)	0.8533293	21648
Schindler’s List (1993)	0.8509731	23193
Satan’s Tango (SÃ¡tÃ¡ntangÃ³) (1994)	0.8500199	2
Casablanca (1942)	0.8078507	11232

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

Regulization with movie + user effect

We will divide the edx data into train and test data, and at the end we will use the validation data.

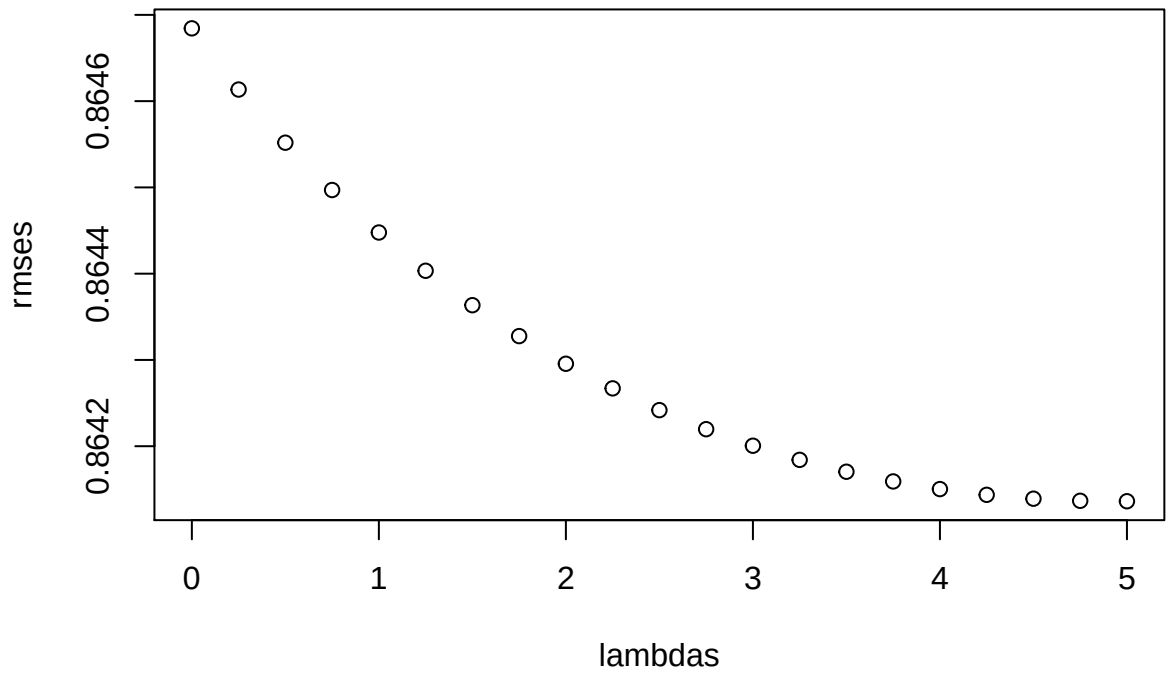
```
# preparing the training data and test data the will use cross validation to determine the best lambda.

set.seed(1, sample.kind="Rounding")
```

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

```
test_index_Reg <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
edx_Reg <- edx[-test_index_Reg,]
temp_Reg <- edx[test_index_Reg,]


test_Reg <- temp_Reg %>%
  semi_join(edx_Reg, by = "movieId") %>%
  semi_join(edx_Reg, by = "userId")



removed_Reg <- anti_join(temp_Reg, test_Reg)
```

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

```r
edx_reg <- rbind(edx_Reg, removed_Reg)

rm(test_index_Reg, temp_Reg, removed_Reg)

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

lambdas <- seq(0, 5, 0.25)

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


  mu <- mean(edx_Reg$rating)


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


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


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

  return(RMSE(predicted_ratings, test_Reg$rating))
})

plot(lambdas, rmses)
```




```
lambda <- lambdas[which.min(rmses)]
paste('Optimal RMSE of',min(rmses),'is achieved with Lambda',lambda)
```

```
## [1] "Optimal RMSE of 0.864136212594715 is achieved with Lambda 5"
```

```
# Now we will use the lambda to calculate the RMSE using the Validation data

pred_y_lse <- sapply(lambda,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 #validation

  return(predicted_ratings)

})
finalRMSE = RMSE(pred_y_lse, validation$rating)
```

```
## Warning in bind_rows_(x, .id): binding character and factor vector,
## coercing into character vector
```

Results

The results we have obtained form different modules are shown in the below table.

method	RMSE
Just the Average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effect Model	0.8653488
Regulized Movie Effect Model	0.9438566
Regulized Movie + User Effect Model	0.8648177

Conclusion

In this recommendation project our goal was to predict the movies rates for specific user, and the challenge was to achieve RMSE value less than 0.8649. We have started our analysis by observing the and analyze it; then we have started our predictions by predicting the average value for all movies rates by all users, and this as expected resulted with a high RMSE value, and to improve our predictions we calculated the average difference between the "rates for each movie" and the "total average", the formula is average of [b_i = rate - mu] , and called the difference "movie bias", and applied that value in our calculations on the validation data, the results got slightly improved due to two main factors, the first factor is that we neglected the user effect in rating the movie, and the second factoe is that there is alot of movies has a little number of ratings and the got the highest and lowest biases values which corrupted the calculations.

Now to fix the first issue, we have added the user effect in our calculations, we calculated the average difference between the "user rates for each movie" and the "movie bias" and the "total average", the formula is average of [u_i = rate - b_i - mu]; we noticed that we have got much bettr results and we are almost near our goal.

And to fix the second issue, we had to add a penalty value (lambda) that will mitigate the less known movies effect on the calculations, this called the regulization model, we divided our data into train and test data to use the cross validation and select the best value of lambda. we applied regulization on both movie and user effect and select the lambda and use it in our final model and applied it on the validation data, and finally achived our goal by having the RMS less than 0.8649.