Recomendation System Final Report

Victor Guerrero

28/03/2020

Overview

In this report I detail the process by which a Recommendation System is created based on the expected rating that each user will give a certain movie. The code below creates the Data Set.

Create edx set, validation set ##################################

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

```
## Loading required package: tidyverse
```

```
## -- Attaching packages ---------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
## Warning: package 'forcats' was built under R version 3.6.3
```

```
## -- 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
```

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

```
## Warning: package 'lattice' was built under R version 3.6.3
```

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

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

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

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

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

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

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

ratings <- fread(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")
```

Validation set will be 10% of MovieLens data

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

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

if using R 3.5 or earlier, use `set.seed(1)` *instead*

```
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 set

```
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)
```

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

```
 edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

Two data sets are created One "edx" which contains 90% of all data, (train set), and the "validation" with only 10% of the data (test set). I will now focus on describing the "edx" Data set.

The "edx" Data set created named consists of 9000055 rows and 6 Columns, the columns are, userId, movieId, rating, timestamp, title, genres.

User Id is the identification of the person giving the rating. MovieID is identification of the movie. Rating is the rating given by the User to a specific movie. Timestamp is the date in which the rating was given. Title is the Title of the movie and genres are the categories in which the movies are classified.

Further analysis shows that there are 10677 movies rated from 69878 users. There are no N/As and there aren't any 0 ratings. The "edx" data set is of class data frame. Additionally, users rate movies at different frequencies, since a variation in both ratings per user and ratings per movie was observed.

Method

Various methods were considered including Linear Regression and Knn nearest neighbours, but both seemed impractical due to variabilities in the frequency in which movies are rated and variation in the number of ratings the movie has. Furthermore, since the data frame is so large even with the bootstrap approach the estimates did not yield positive results. As a result, I decided to use a mean/ data regularization approach, taking into consideration user and movie bias of tackling this issue.
To evaluate the performance, the Residual Sum of Squares or RMSE was used. Which was created by this code.

Create a function that calculates de RMSE

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

First, we need to confirm that there is User and Movie Bias.To calculate this the first thing to be done is find the mean of all movies. That was done with the code below.

Calculate the mean rating of all movies

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

Next we find the bias per movie with the code below.

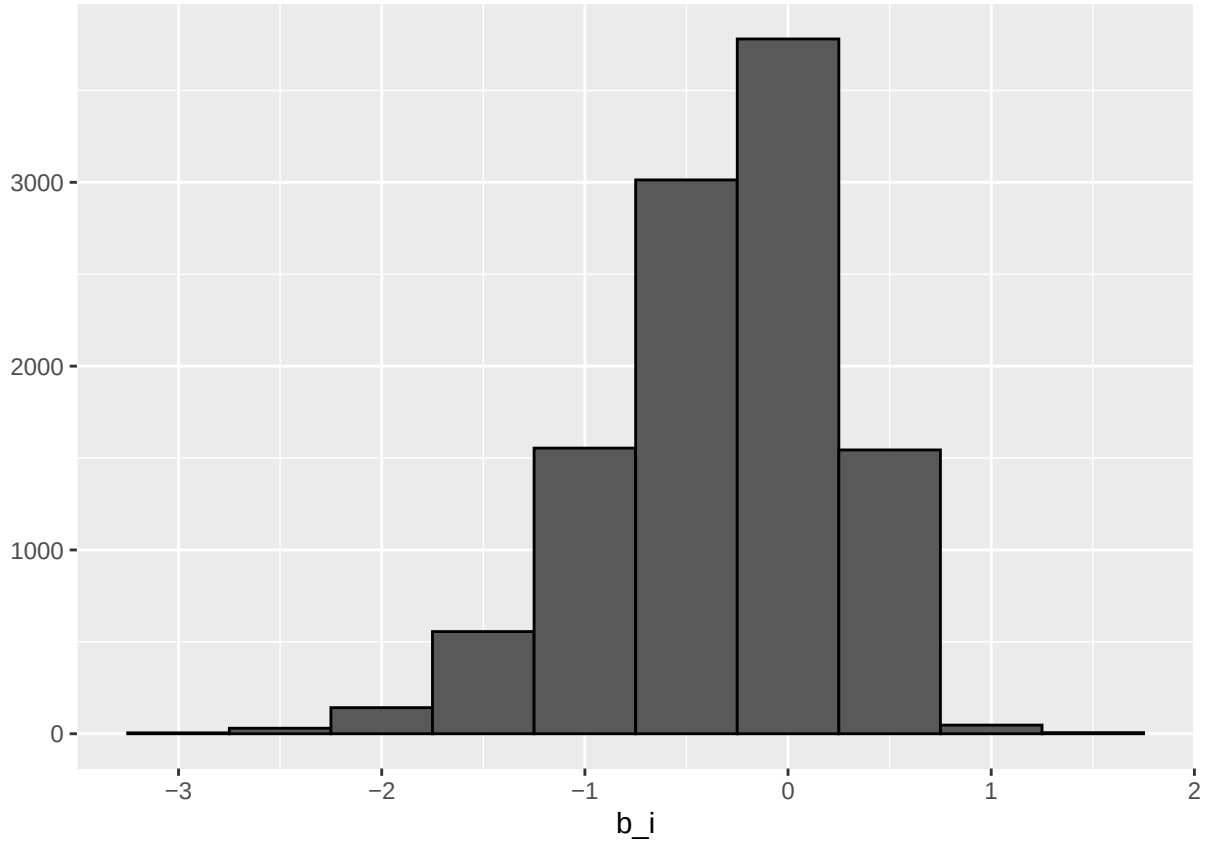
Calculate the Bias per Movie

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

As you can see the bias is obtained by subtracting the average of all movies to the average rating per movie. The bias can be visualized by the code below.

Confirming User Movie Bias

```
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"))
```



This yields the following plot. As you can see most of the movies conform to the average thus having a bias of 0, however less than 4 thousand movies conform to the average, the rest having bias mostly between 1/-1 rating.

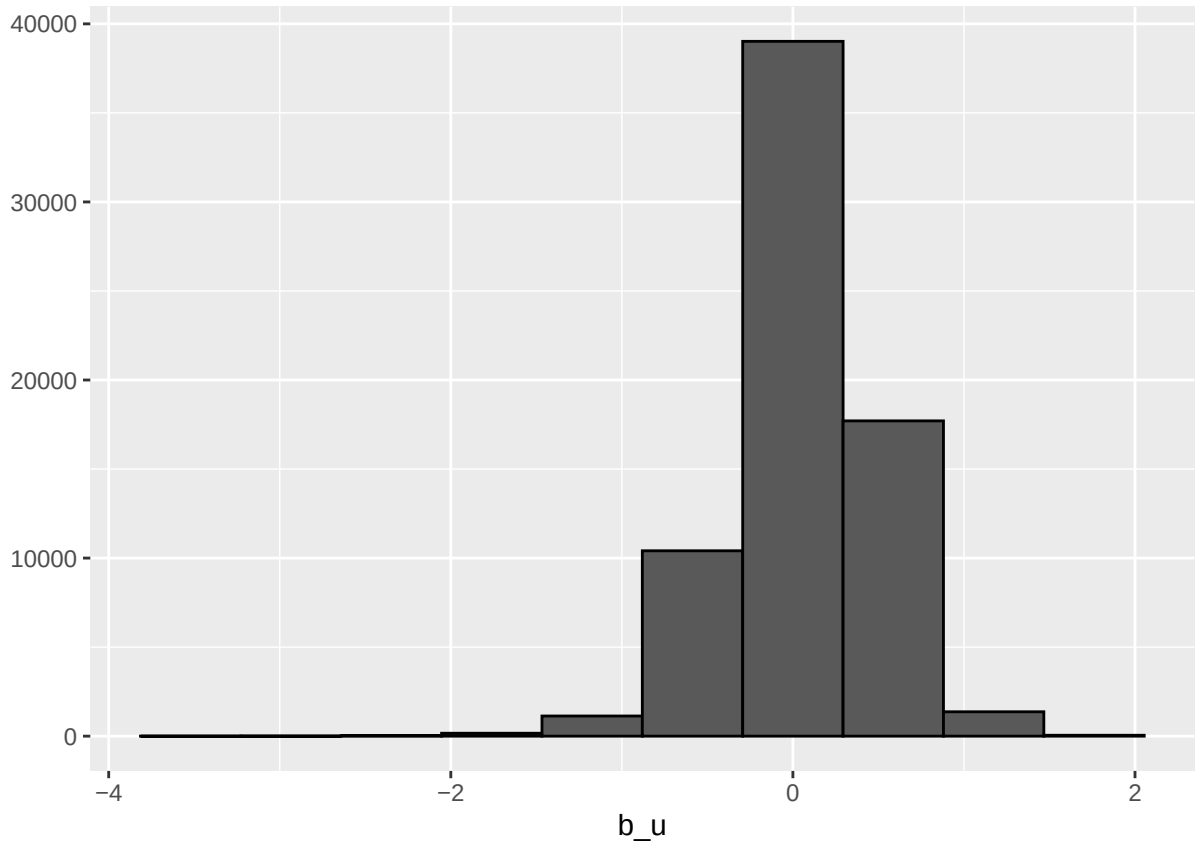
Next we will account for User Bias. Which is found and visualized with the below code.

Calculate User Bias

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

Confirming User Bias

```
user_avgs %>% qplot(b_u, geom ="histogram", bins = 10, data = ., color = I("black"))
```



Which gives us the following plot.

Although less biased (as more conform to the norm), bias can still be detected.

Now that the bias per movie and user is confirmed we proceed to evaluate our code based on the findings, accounting for movie and user bias. See code below.

Create Vector of predictions

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

However, we find that the RMSE is too high or 0.87. By using this code.

Find RMSE

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

```
## [1] 0.8653488
```

(Please note that this step is not included in the code file, as its just to explain the procedure by which I arrived to the targeted result.)

To arrive to the target result I used regularization by which I give a weight to the ratings based on how often it was rated thus, reducing the error produced by them.

To do this first I determined the best value for Lambda. Using the code below.

Creation of lambdas list

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

Regularization of Bias and determining best Lambda Value

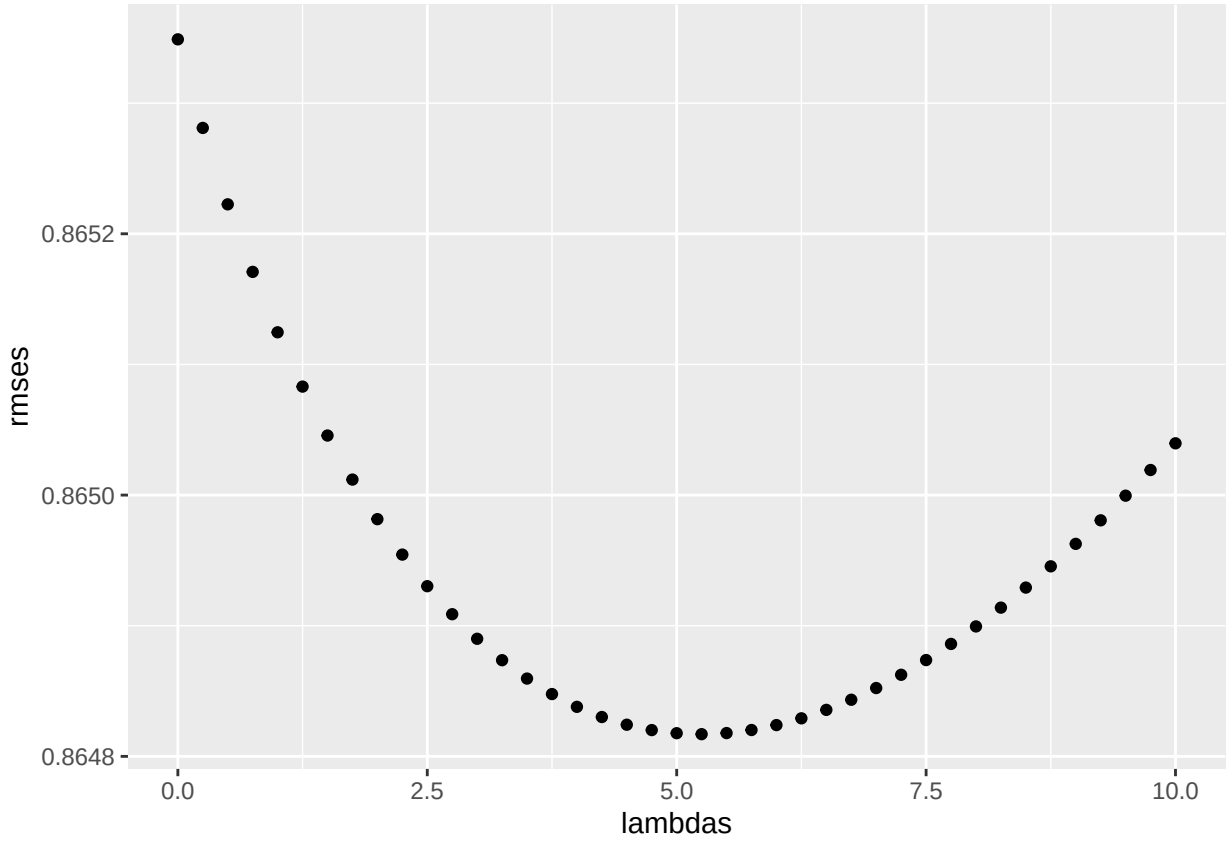
```
rmses <- sapply(lambdas, function(l){
            mu <- mean(edx$rating)
            b_i <- edx %>%
                      group_by(movieId) %>%
                      summarize(b_i = sum(rating - mu)/(n()+l))
            b_u <- edx %>%
                      left_join(b_i, by="movieId") %>%
                      group_by(userId) %>%
                      summarize(b_u = sum(rating - b_i - mu)/(n()+l))
            predicted_ratings <-
                      validation %>%
                      left_join(b_i, by = "movieId") %>%
                      left_join(b_u, by = "userId") %>%
                      mutate(pred = mu + b_i + b_u) %>%
                      .$pred
            return(RMSE(predicted_ratings, validation$rating))
        })

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

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

The lambda object is then visualized since the lambda values where arbitrary based on other previous successful projects. The lambda can be visualized with the following code.

Plot lambda values

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



As you can see the lambda values are well chosen as after 5.25 they start going up exponentially.

Results

Now that we have everything in place it is time to calculate the result now with lambda determined. See code below.

Rmse calculation with optimized lambda which results in 0.864817

```
Final.RMSE <- mu <- mean(edx$rating)
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+lambda))
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+lambda))
  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred

RMSE(validation$rating, predicted_ratings)
```

```
## [1] 0.864817
```

Left join is used to create the list of predicted ratings so that the variables b_i, b_u and mu match validation set. As you can see with regularization the target of below 0.8649 is met.

Conclusion

A reliable recommendation can be created, based on the history of ratings given by the user and the popularity (average rating of the movie). Movies with higher ratings will be recommended more often.

It is still a simple model, and it could probably be improved by integrating the genres bias into the formula. Finally, users tend to prefer movies of similar genres which is an indication of further improvement that could be done to the algorithm.