Predicting Movie Ratings from User, Movie, and Genre effects in Loss Functions using the MovieLens 10M Dataset

In partial fulfillment for the Data Science Professional Certification from Harvard University

Arturo P. Caronongan III

July 16, 2020

Abstract

This project is the first of two deliverables for Harvard University's Data Science Professional Certification under the course PH125.9x Data Science: Capstone. In this project, the MovieLens 10M dataset will be analyzed to build a Movie Ratings Prediction Module for Movie Recommendations generated from computational models using both non-regularized and regularized loss functions.

Contents

1.0 Introduction and Overview of the Problem

The growth of today's development in technology, the rapid growing contents in multimedia libraries and the digital industries have called for a need of effective information filtering systems, and in particular recommendation systems. In the case of the growing movie industry, the amount of movie content available is increasing the need to efficiently access, discover, and to present it to the final user. In order to answer this need of delivering relevant information to the user, movie recommendation systems are developed to produce movie lists that cater to the user's wants.

The main purpose of a movie recommendation systems is to estimate the user's preference and present them with items that may fit their preference.[2,5] This is because users normally do not have the time to search through these collections looking for new items due to the huge amount of movie contents available. In order to come up with the most suitable recommendation, techniques for classifying, analyzing and retrieving audio content information have to be conducted.

Netflix uses a recommendation system to predict how many stars a user will give a specific movie where low-rated movies indicate that it is not a good movie for the user while highly rated movies are considered as suitable for the user's preference.[4] Items which are then predicted as movies that will receive a good rating based on the user's history of ratings are then recommended to the user. As a result, Netflix hosted a challenge that called for algorithms that would improve their recommendation model's approach. The winning solution made use of a Loss function that made use of the Residual Mean Squared Error (RMSE) in calculating the prediction's performance.[3] More about this approach will be discussed later.

For this problem, an approach that is similar to the Loss function used will be applied to produce a similar predictive model for movie recommendations using a subset of the dataset used, the dataset being called the MovieLens 10M Dataset.[1]

1.1 MovieLens 10M Dataset

The MovieLens dataset is a ubiquitous data that is comprised of "26,000,000 ratings and 750,000 tags applied to 45,000 movies by 270,000 users". [1]

For this report, a subset of the complete dataset will be used. This subset, aptly called MovieLens 10M - named for being 10 million records long - will undergo data analysis and used as a basis for predicting ratings of movies based on current ratings provided by users.

1.2 Data Preparation - Loading the MovieLens Dataset

To begin, all the data was split into two subsets, namely the training dataset named `edx`, and the validation dataset named `validation`.

The script to load the dataset are as follows.

```
################################
# Create edx set, validation set
#
# - Provided by edx
################################

# Note: this process could take a couple of minutes

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

# 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")
```

As indicated in the next code, edx data consists of 90% of the MovieLens dataset while the validation data consists of 10% of the MovieLens dataset. The `edx` data set will be used to build our models used for predicting movie values, while the `validation` data set will be used to evaluate the models that were developed.

```
# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
# 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)
edx <- rbind(edx, removed)

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

2.0 Preliminary Data Analysis

In this section, we are going to take a closer look at the dataset that has been provided. Before we can properly apply algorithms for predicting movie ratings, it is important that we are aware as to how our data is structured, insights we can obtain from our data, and the number of entries available in the dataset.

For this entire section, as we are dealing with the `edx` dataset for training, our commentary for this section will focus mostly on the `edx` dataset.

2.1 Analyzing the instances of ratings in the dataset

First, it is important to know how many ratings we have in the dataset and how each rating entry is labelled out. We can see that using the following script.

```
## [1] 9000055       6
```

We can see that we have 9000055 rows and 6 in the data set. To give us an idea of how the data is laid out, we can illustrate the first few entries of the dataset using the following script.

```
##   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
```

This gives us a better idea of how our data is structured out. The 6 given column headers are as follows:

`userID` - numeric (denotes a distinct user entry)
`movieId` - numeric (denotes a distinct movie entry)
`rating` - numeric (denotes the rating userId gave to movieId)
`timestamp` - numeric (denotes the date the rating was made in seconds)
`title` - character (denotes the title of the movie with ID movieId)
`genres` - character (denotes the genre label of the specific movie)

We can see that each row actually denotes a rating entry provided by a user. We could analyze the dataset to see how many distinct users and distinct movies there are.

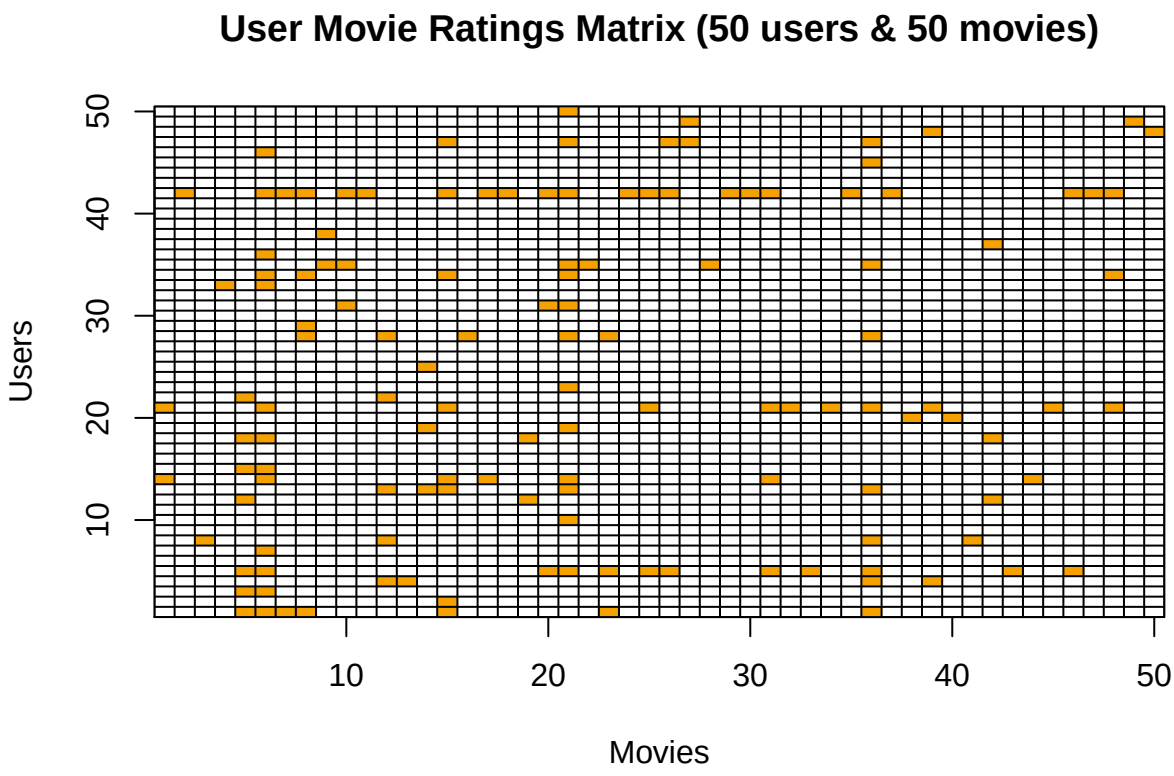
```
##   n_users n_movies
## 1   69878    10677
```

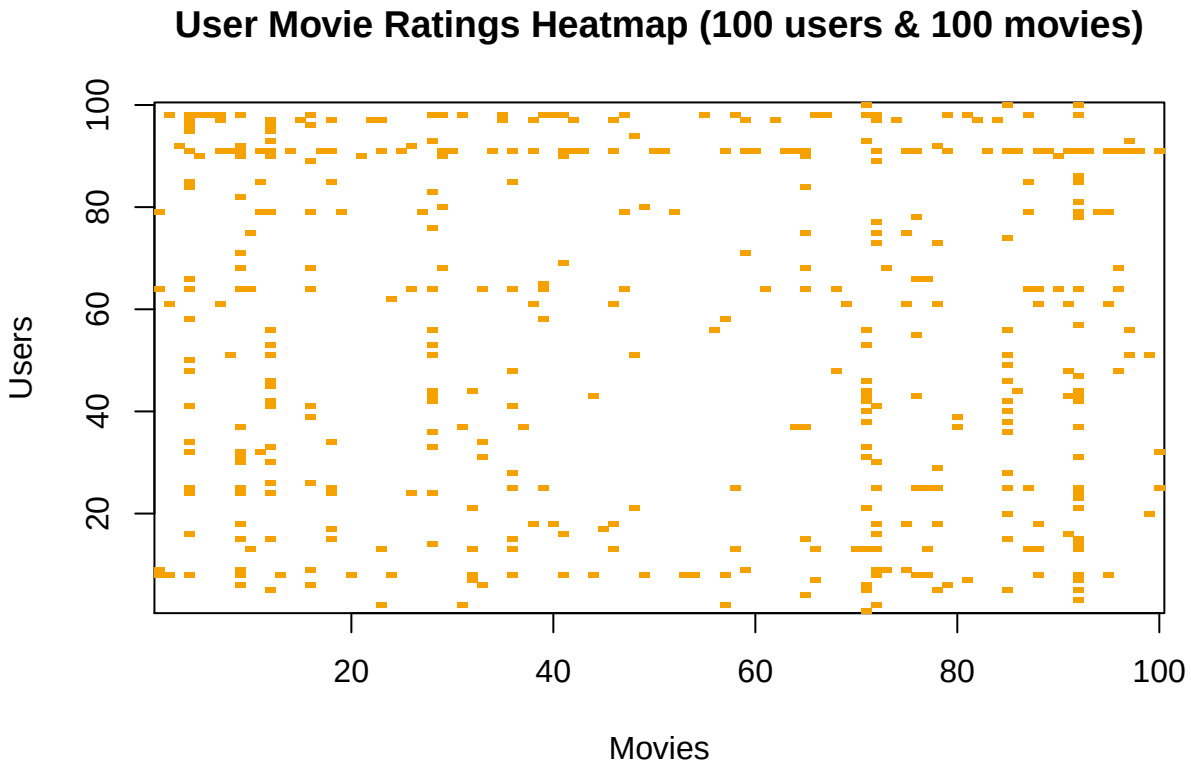
Given that there are 69878 users and 10677, the best case would indicate that all users have given a rating for all movies. Unfortunately, this is not the case as that would indicate that our dataset should have 746087406 rows compared to the 9000055 values that our dataset has.

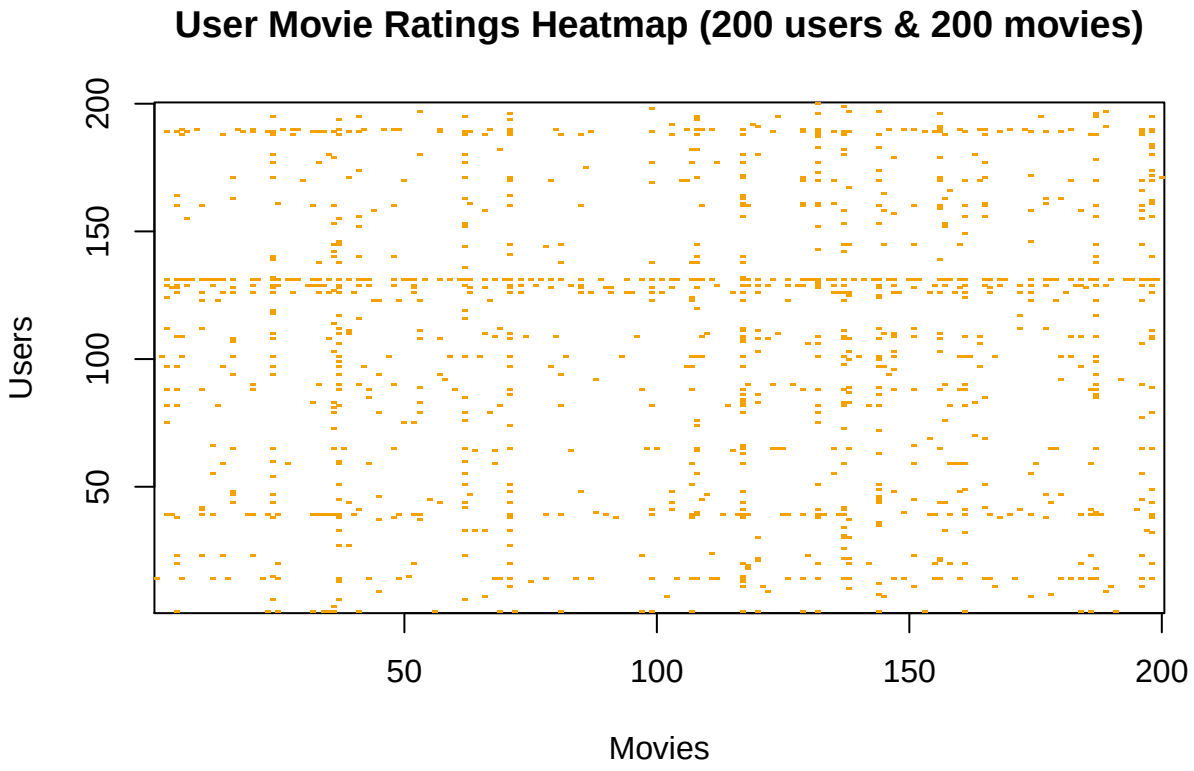
```
##   total_complete_ratings available_ratings percentage
## 1              746087406           9000055   0.012063
```

This means that we have approximately only 1.2063004% of overall ratings available, indicating a sparse matrix of ratings.

To give a clearer idea of the matrix, we can visualize the user to movie ratings matrix. In the following images, we can clearly see that our data matrix is sparse.




User Movie Ratings Heatmap (100 users & 100 movies)
Users
20
40
60
80
100
Movies
20
40
60
80
100

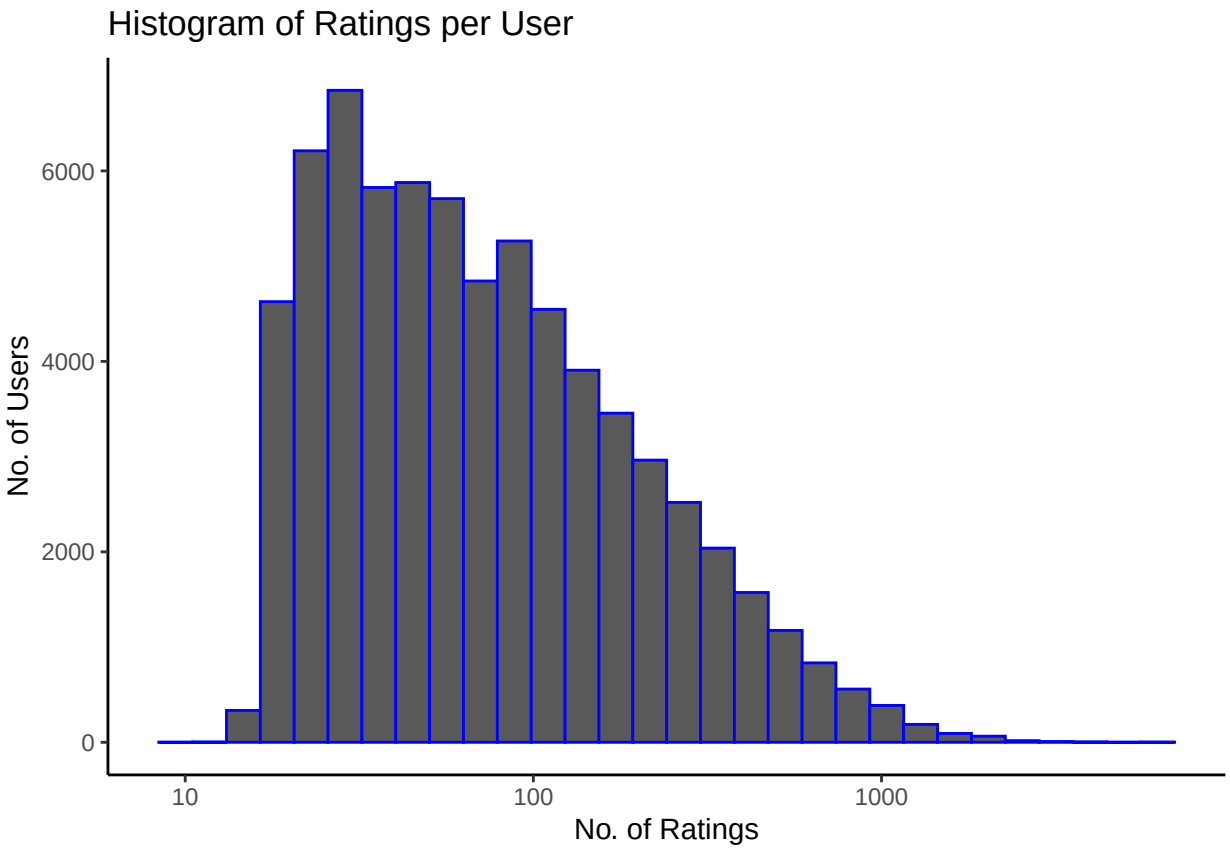


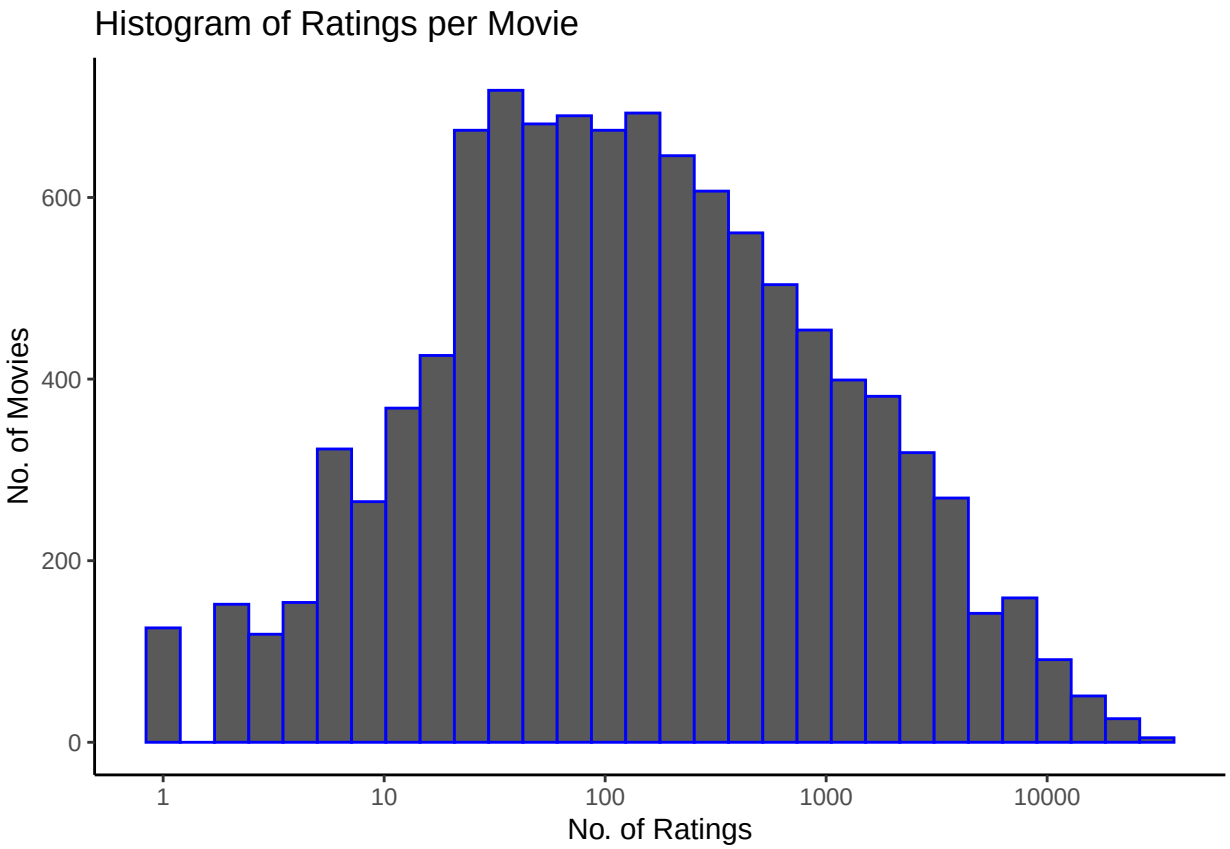
Given the sparse matrix, we need to come up with a method to be able to predict those blank ratings. Nonetheless, we can observe that different users seem to have different behaviors in terms of rating movies. It is therefore important to analyze the users effect in terms of rating movies as a basis for recommendation.

2.2 Analyzing Ratings per User and per Movie

Given the analysis from the previous module, the next process of analysis comes with analyzing the distribution of ratings amongst users and movies in the dataset. Examining the histogram of users and movies respectively, we can come up with the following insights:

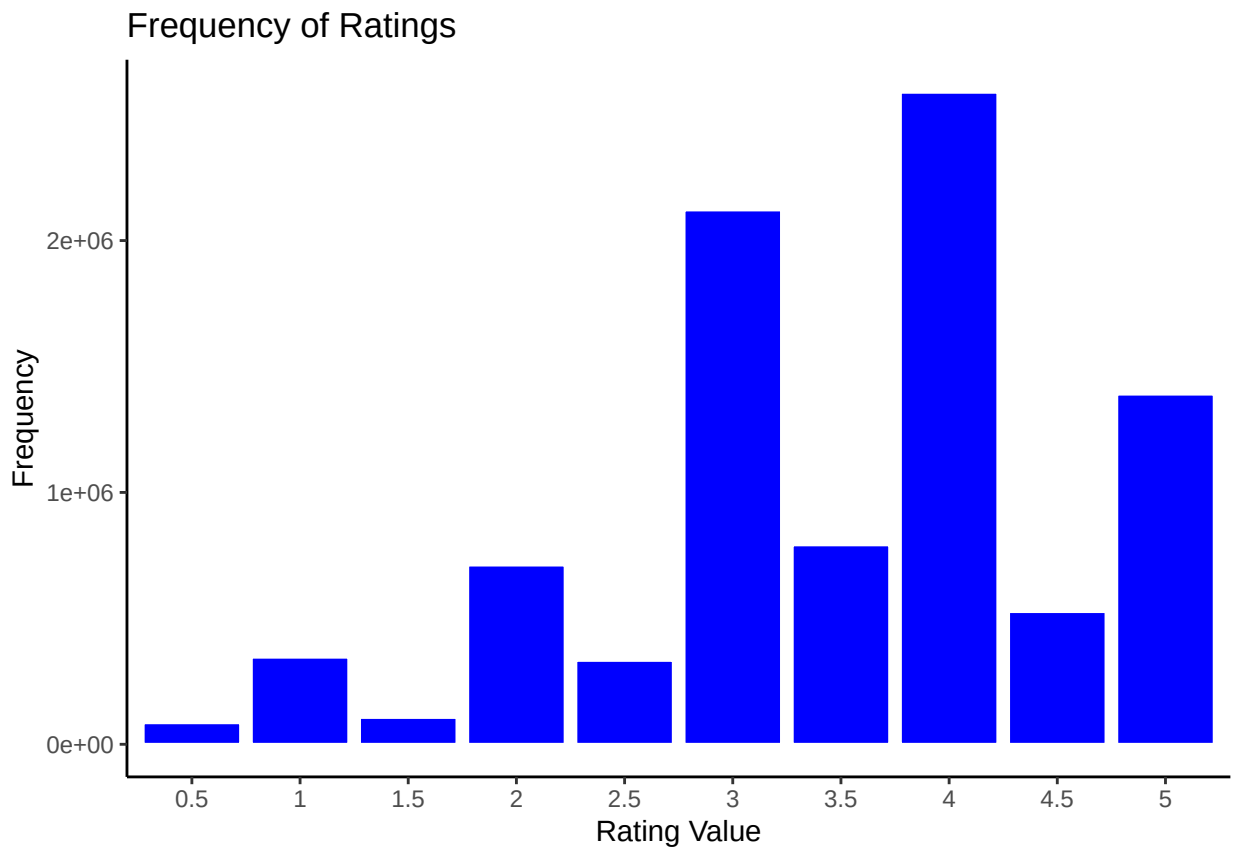
- There are some users who give more ratings than other users, however, majority of the users tend to give relatively few ratings
- We can see that some movies are rated more frequently than others, some of which have a very small amount of ratings


Histogram of Ratings per User
No. of Users
6000
4000
2000
0
10
100
1000
No. of Ratings



To give a better look at the reviews that were provided by the users, we can refer to the following table and chart.

```
## # A tibble: 10 x 2
##    rating   count
##     <dbl>   <int>
##  1    0.5   85374
##  2    1    345679
##  3    1.5  106426
##  4    2    711422
##  5    2.5  333010
##  6    3   2121240
##  7    3.5  791624
##  8    4   2588430
##  9    4.5  526736
## 10    5   1390114
```




```
##   ave_rate stdev_rate
## 1     3.51       1.06
```

We can see from the information that the average ratings amongst all movies is `3.51` with a standard deviation of `1.06`. Most movies are given a rating of `4`. It can also be observed that users normally gave whole number ratings in lieu of decimal value'd ratings.

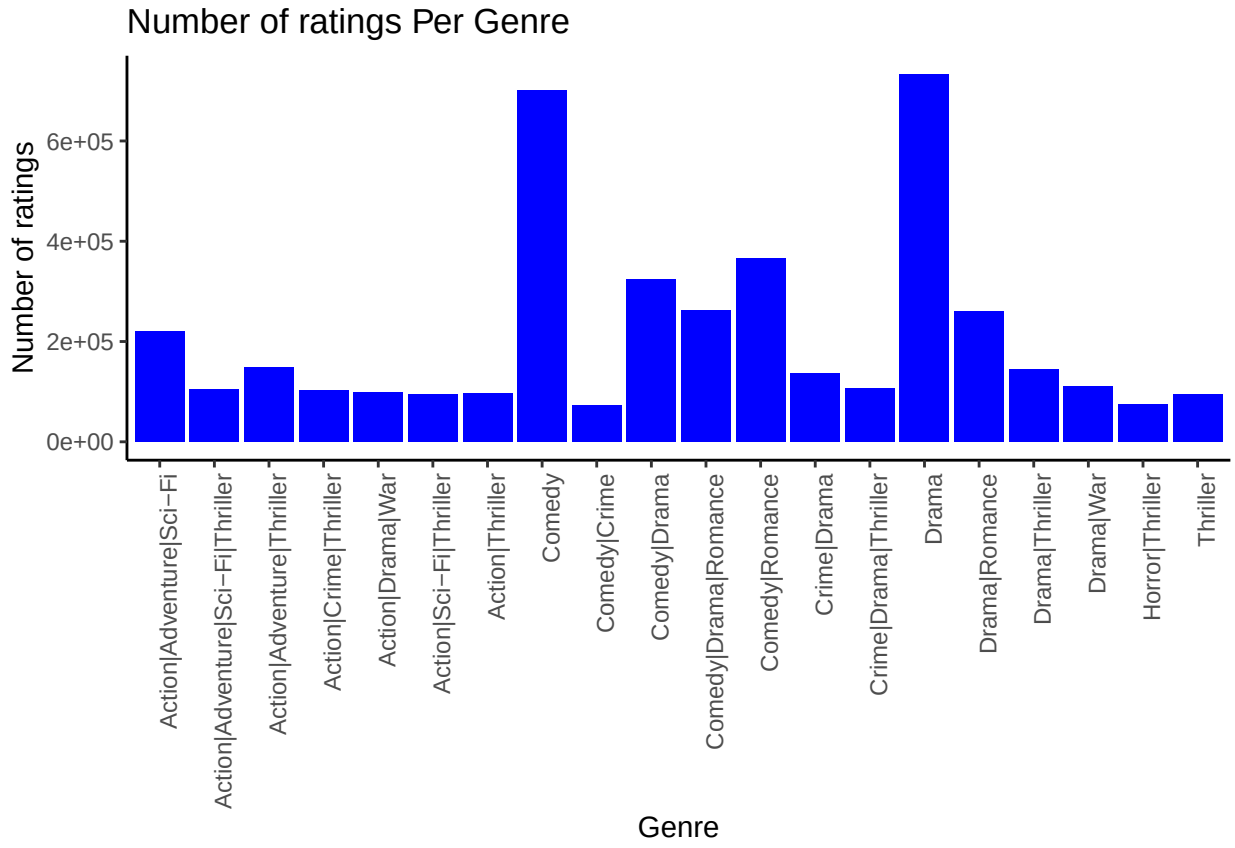
2.3 Analyzing Ratings per Movie Genre

It can be inferred that genres can have an impact on what a movie's rating will be given different user's preferences on movie types. Here, we can observe that, while movies can be tagged with multiple genres bringing the total number of genre listings to 797, the data set contains 20 unique genres.

```
##  [1] "Comedy"             "Action"             "Children"
##  [4] "Adventure"          "Animation"          "Drama"
##  [7] "Crime"              "Sci-Fi"             "Horror"
## [10] "Thriller"           "Film-Noir"          "Mystery"
## [13] "Western"            "Documentary"        "Romance"
## [16] "Fantasy"            "Musical"            "War"
## [19] "IMAX"               "(no genres listed)"
```

We can observe which genres & combinations are most often rated. This can definitely play a role in determining which types are most watched and most likely preferred by users.

```
## # A tibble: 20 x 2
##    genres                              count
##    <chr>                               <int>
##  1 Drama                              733296
##  2 Comedy                             700889
##  3 Comedy|Romance                     365468
##  4 Comedy|Drama                       323637
##  5 Comedy|Drama|Romance               261425
##  6 Drama|Romance                      259355
##  7 Action|Adventure|Sci-Fi            219938
##  8 Action|Adventure|Thriller          149091
##  9 Drama|Thriller                     145373
## 10 Crime|Drama                        137387
## 11 Drama|War                          111029
## 12 Crime|Drama|Thriller               106101
## 13 Action|Adventure|Sci-Fi|Thriller   105144
## 14 Action|Crime|Thriller              102259
## 15 Action|Drama|War                    99183
## 16 Action|Thriller                     96535
## 17 Action|Sci-Fi|Thriller              95280
## 18 Thriller                            94662
## 19 Horror|Thriller                     75000
## 20 Comedy|Crime                        73286
```



From the chart above, we could see that Drama and Comedy are the most rated genres. We can perform further analysis and identify what genre combinations lead to the highest ratings.

```
## # A tibble: 10 x 2
##    genres                                    mean_rating_by_genre
##    <chr>                                                    <dbl>
##  1 Animation|IMAX|Sci-Fi                                     4.71
##  2 Drama|Film-Noir|Romance                                   4.30
##  3 Action|Crime|Drama|IMAX                                   4.30
##  4 Animation|Children|Comedy|Crime                           4.28
##  5 Film-Noir|Mystery                                        4.24
##  6 Crime|Film-Noir|Mystery                                  4.22
##  7 Film-Noir|Romance|Thriller                               4.22
##  8 Crime|Film-Noir|Thriller                                 4.21
##  9 Crime|Mystery|Thriller                                   4.20
## 10 Action|Adventure|Comedy|Fantasy|Romance                  4.20
```

Likewise, if we are to take a look at the lowest rated genre combinations, we can infer which genres possessed the lowest ratings.

```
## # A tibble: 10 x 2
##    genres                                      mean_rating_by_genre
##    <chr>                                                      <dbl>
##  1 Documentary|Horror                                          1.45
##  2 Action|Animation|Comedy|Horror                              1.5
##  3 Action|Horror|Mystery|Thriller                              1.61
##  4 Comedy|Film-Noir|Thriller                                   1.64
##  5 Action|Drama|Horror|Sci-Fi                                  1.75
##  6 Adventure|Drama|Horror|Sci-Fi|Thriller                      1.75
##  7 Action|Adventure|Drama|Fantasy|Sci-Fi                       1.90
##  8 Action|Children|Comedy                                      1.91
##  9 Action|Adventure|Children                                   1.92
## 10 Adventure|Animation|Children|Fantasy|Sci-Fi                 1.92
```

We can see that the prevalence of the Horror genre in the lowest and it's absence in the highest rated shows slight consistency in the lack of ratings. We can also hypothesize that there is an effect on genre with regards to user preference in terms of rating movies.

We can definitely see that there is a user effect and a genre effect with regards to how the ratings are affected.

3.0 Modeling Predictions through the Loss Function

The function that computes the RMSE of corresponding predictive models given a vectors of ratings will be the following equation.

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

Where: $y_{u,i}$ is defined as the rating for movie i by user u, $\hat{y}_{u,i}$ is defined as the prediction for movie i by user u, and N is defined as the total number of user/movie combinations and the sum occurring over all these combinations.

Under these circumstances, the RMSE can be interpreted similarly to the standard deviation, where it is the typical error made when predicting a movie rating. An RMSE that is larger than 1 indicates that the error is off by 1 star rating than what the user would tend to give, so the aim is to have a model that can produce an RMSE that is as low as possible.

3.1 Baseline Loss Function Model

The first model is going to be used as the Baseline Model, which will operate under the consideration that all movies will be rated equally regardless of the movieId and userId attributes. The linear model that assumes the same rating for all movies and users with added random variations set is computed as follows.

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

With $\epsilon_{u,i}$ considered as the independent errors from the same distribution centered at the value of 0 and μ being considered as the "true" rating for all movies. Given the following circumstances, the value that would minimize the RMSE is the least squares estimate of μ which can be derived as the average of all ratings.

We can generalize the model by obtaining the average of our training dataset (which, as previously mentioned, is 3.5124652) and use it to calculate the first RMSE obtained

```
mu <- mean(edx$rating)
baseline_rmse = RMSE(validation$rating, mu)
baseline_rmse
```

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

As indicated previously, the goal is to have a low RMSE so an RMSE > 1 means that ratings are normally inaccurate by 1 star. Given the baseline, we look to improve it by using the insights that we have obtained through data analysis by adding the Movie Effect, User Effect, and Genre Effect.

3.2 Modeling Movie Effect in the Loss Function Model

We know from experience that some movies are rated better than other movies so for our next model, we will take that into consideration. It is a given that blockbuster movies are normally watched by a wider array of audiences and lead to more and typically higher ratings. As a result, we could consider adding the term b_i (bias) to represent the average ranking for movie i.

As a result, the resulting linear equation will look as follows:

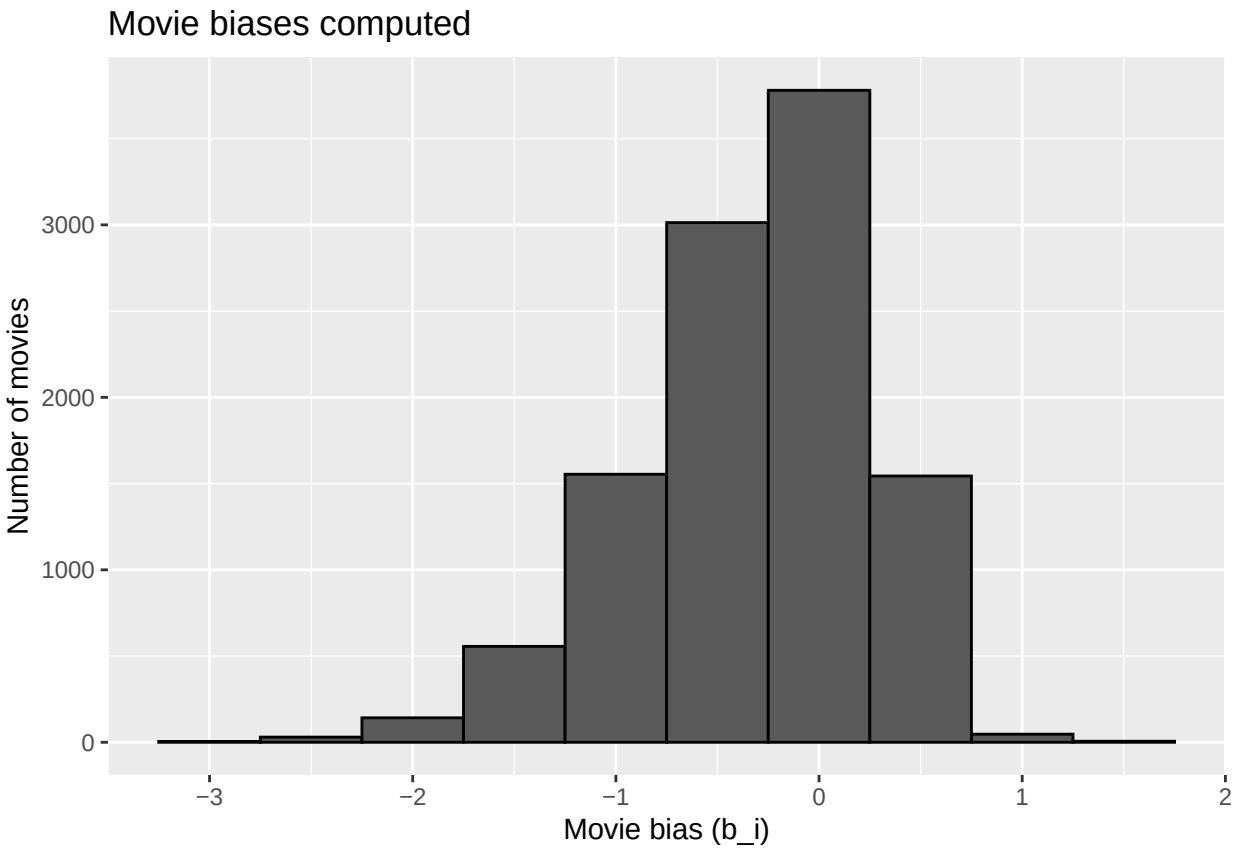
$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

It can be inferred that the least squares estimate b_i is the average of $Y_{u,i} - \mu$ so the bias values can be obtained through the following code segment.

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

Plotting the obtained biases of each movie, it can be observed that the obtained biases vary greatly as shown in the succeeding figure.

```
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"),
                     xlab = "Movie bias (b_i)",ylab = "Number of movies",
                     main = "Movie biases computed")
```



It can also be observed that, although a sizeable amount of movie ratings fall within the average rating (a bias of 0), there are several movies that have performed considerably worse than average, and that could have affected the baseline model negatively.

Plugging in the movie biases into the prediction model leads to an improvement in the RMSE as observed in the following code segments. First, the predicted ratings are then generated with the new model through the following script.

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

Calling a similar script by computing for the RMSE between the newly predicted ratings and those of the validation data set.

```
movie_effect_rmse <- RMSE(predicted_movie_effect_ratings, validation$rating)
movie_effect_rmse  #Compared with the validation dataset
```

```
## [1] 0.9439087
```

We can see that there is a slight improvement of the RMSE obtained compared to that of the Baseline Loss Model. The slight improvement is due to the predictions taking into consideration the fact that movies are rated differently by adding the computed b_i to μ, in this case, the average rating of all movies. If an individual movie is, on average, rated worse than μ , the model predicts that it will rated lower than μ by b_i, which in turn reflects the difference of the individual movie's average rating from that of the overall average.

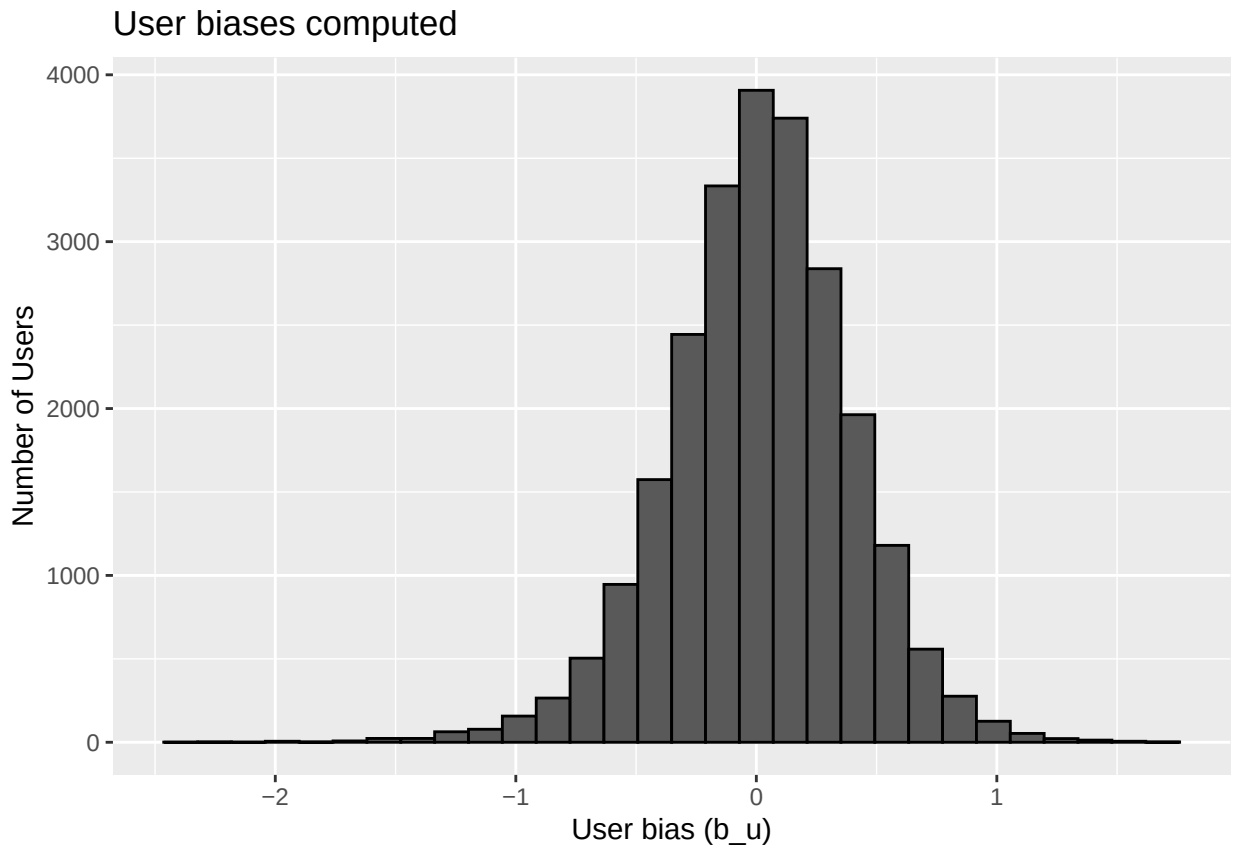
A limitation of this model is that it does not consider users and the individual user's effect to ratings. We noticed that different users have different behaviors when it comes to rating movies, which can depend on

the preference of the user, how much time a user is willing to invest in rating movies, or how much of a movie enthusiast the user is. As a result, we could further improve on the model to take into consideration the user effect.

3.3 Modeling Movie and User Effect in the Loss Function Model

Similar to the Movie Effect, the biases of each particular user is obtained. However, it should be noticed from the pre-analysis stage that not all user have a sufficient amount of reviews to make a generalization, so this study will place a threshold on considering users that have placed at least 100 reviews.

```
user_biases <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating - mu - b_i))

user_biases%>% qplot(b_u, geom ="histogram", bins = 30, data = ., color = I("black"),
            xlab = "User bias (b_u)",ylab = "Number of Users",
            main = "User biases computed")
```



User variability is evident in the graph. It can be seen that some users have a negative impact on a movie's rating while others contribute positively towards a movie's rating.

We can add the corresponding user bias b_u into our model to arrive at the following equation.

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

where b_u is a specific to a particular user. The effect of this allows a better estimate of how a particular user would rate a great movie (A movie that is commonly rated as a 4 is rated as a 2 due to the user's high standards as an example) so the model is able to have a greater ability to predict whether a particular user will have the tendency to underrate a particular movie and vice-versa.

In a similar fashion to that of the movie effect, we can obtain the following approximation by computing μ and b_i, thus estimating b_u as the average of $Y_{u,i} - \mu - b_i$.

The following code segment demonstrates the procedure.

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

Like generating predicted movie ratings with added movie effect, the following code segment is used to generate predicted movie ratings taking in the user preferences and generated user bias b_u into consideration.

```
predicted_user_ratings <- validation%>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_effect_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>% ## Observe the given equation above
  pull(pred)
```

As per the previous section, computing for the RMSE between the newly predicted ratings taken from considering the ratings from applying the context of user effect and those of the validation data set leads to a better RMSE.

```
user_effect_rmse <- RMSE(predicted_user_ratings, validation$rating)
user_effect_rmse #Compared with the validation dataset
```

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

3.4 Modeling Movie, User, and Genre effect in the Loss Function Model

It is inevitable that while it's certain that user ratings can be noticed among the blockbuster movies to determine whether a movie is considered good or not, certain users can hold a preference over some genres. An example is that certain users may enjoy Drama and Comedy, but find aversions to Horror movies. As a result of our data analysis, we can see that there is some effect that genre has towards the ratings of a particular movie. This can allow further exposure to lesser known movies but are of the same genre to the blockbusters that have been rated with general approval.

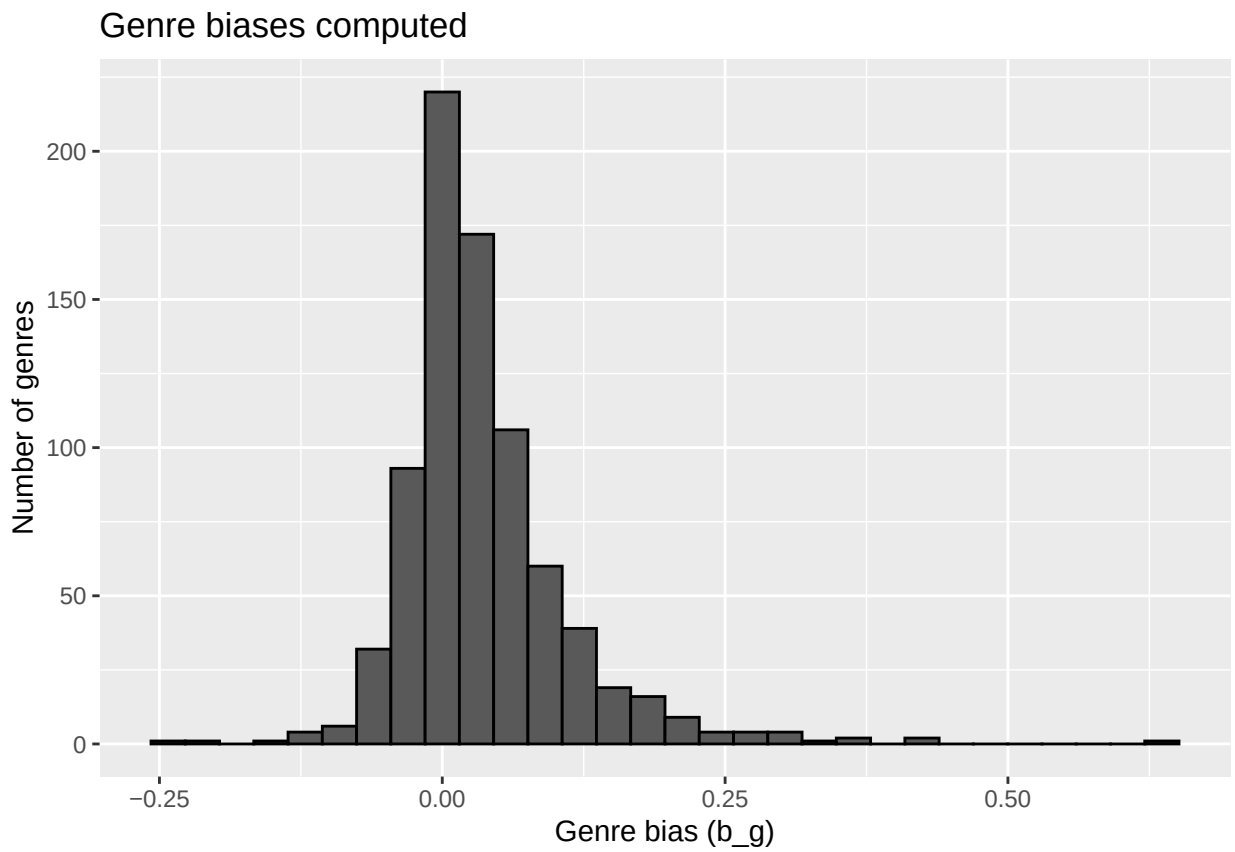
We can do this by further building up to our model and adding the corresponding genre bias b_g into our model to arrive at the following equation.

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

In a similar fashion to that of the movie and user effect, we can obtain the following genre bias approximation by computing μ and b_i, thus estimating b_u as the average of $Y_{u,i} - \mu - b_i - b_u$. Take note, this is due to certain movies being rated differently across varying genres.

We can further visualize the generated b_g that is found in the dataset. As we can see, although majority of the genre combinations are pretty well received, there are some combinations that are just not favorable to the general audience.

```
genre_effect_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_effect_avgs, by = "userId") %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))
genre_effect_avgs%>% qplot(b_g, geom ="histogram", bins = 30, data = ., color = I("black"),
                      xlab = "Genre bias (b_g)",ylab = "Number of genres",
                      main = "Genre biases computed")
```



We can then use the following code segment to generate predicted movie ratings taking in consideration the user preferences generated from considering and generated user bias b_g into consideration.

```
predicted_genre_effect_ratings <- validation %>%
  left_join(movie_avgs, by = "movieId") %>%
  left_join(user_effect_avgs, by = "userId") %>%
  left_join(genre_effect_avgs, by = c("genres")) %>%
  mutate(pred = mu + b_i + b_u + b_g) %>%
  pull(pred)
```

The RMSE should slightly improve as a result of the genre effect that is observed in the dataset. We can clearly see the impact with the slightly improved RMSE obtained.

```
genre_effect_rmse <- RMSE(predicted_genre_effect_ratings, validation$rating)
genre_effect_rmse #Compared with the validation dataset
```

```
## [1] 0.8649469
```

As we can observe from slowly building up our model, we have achieved a lower RMSE by carefully considering multiple variables that exist in the data set, ranging from the user effect, movie effect, and the genre effect.

During data analysis in the earlier section, we observed that we are dealing with a sparse data matrix, meaning there is major imbalance between ratings given as a result of certain users providing several ratings and vice-versa. The true can be said about other movies and genres where certain movies are rated moreso than others. This can produce some noisy estimates, given the RMSE's nature of being sensitive to large errors.

It is due to this that an approach is considered to regualize values by adding penalties to movies that are given a large estimate that is based off of a small sample size.

3.5 Applying Regularization through Penalized Lambda

Regularization is normally a method that is used to reduce the effect of overfitting. In this case, regularization is going to be used to enable the computational model to penalize large values of b_i, b_u, and b_g obtained as a result of being inflated ratings as a result of very few ratings. This is done by adding the penalty value lambda λ to the value of N or the population count. While it should have a smaller impact to those with several consistent ratings (pertaining to a general view of the public regarding that movie), it can be used to reduce the noisy data brought upon by insufficient ratings.

As a result, instead of merely getting the average values to obtain b_i, b_u, and b_g by dividing them by N, we divide them by the value of $N + \lambda$.

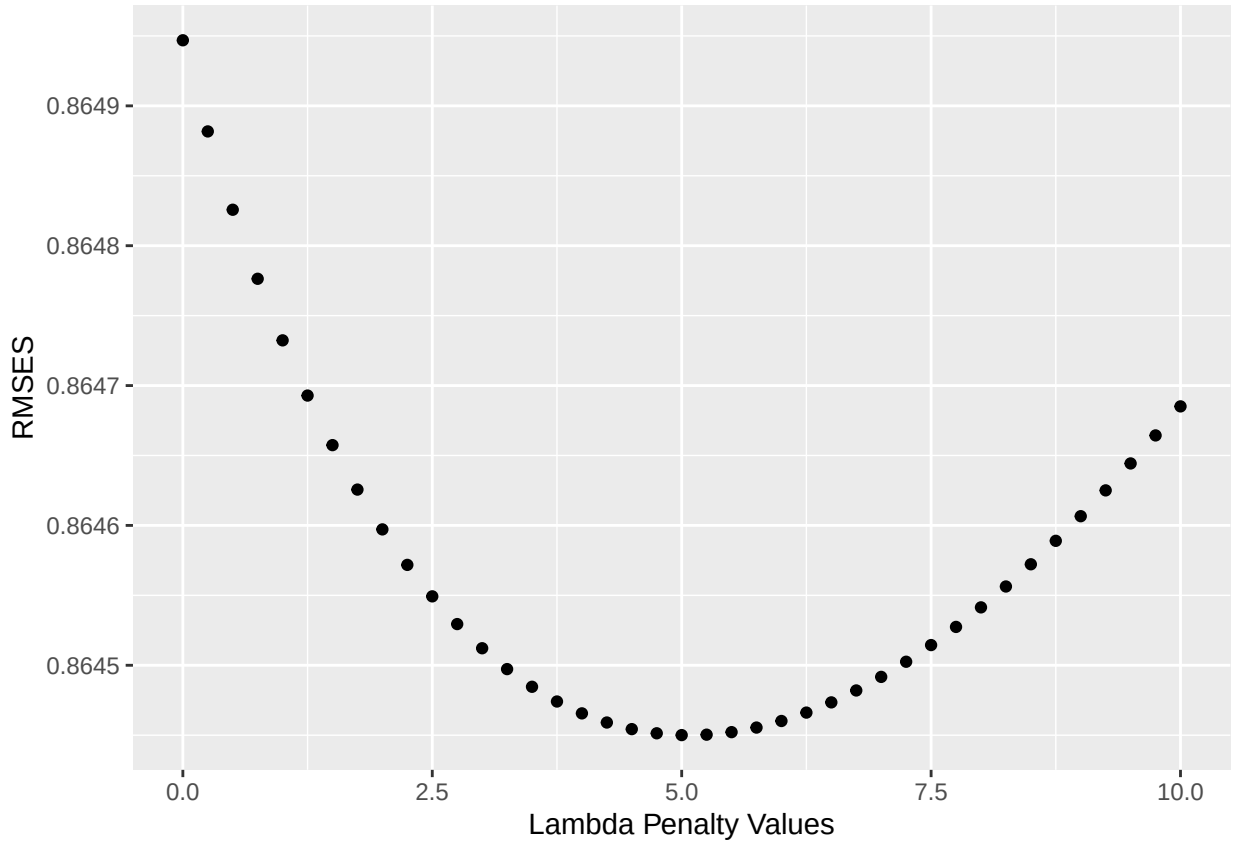
For this, the value of λ ranged from 0.00 to 10.00 inclusively by increments of 0.25. The following code segment demonstrates the regularization process, first by initializing the values of our λ, followed by recomputing the values of our b_i, b_u, and b_g obtained by adding the value of λ to N for each.

```
lambdas <- seq(0, 10, 0.25) ## 0 to 10 inclusively, with increments of .25

rmses <- sapply(lambdas, function(l){ #rmses will be saved in the object called rmses

  mu <- mean(edx$rating)

  b_i <- edx %>% ## Earlier, this was movie_avgs
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l)) ## Divide by N + Lambda

  b_u <- edx %>% ## Earlier, this was user_effect_avgs
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l)) ## Divide by N + Lambda

  b_g <- edx %>%  ## Earlier, this was genre_effect_avgs
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - b_i - b_u - mu)/(n()+l)) ## Divide by N + Lambda

  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
```

```
    left_join(b_g, by = c("genres"))  %>%
    mutate(pred = mu + b_i + b_u + b_g) %>%
    pull(pred)

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

We can see that starting from a penalty λ value of 0, we obtain the RMSE valued from the previous section.This is indicated in the graph that follows, providing the different values of lambdas and the corresponding RMSE obtained.

```
qplot(lambdas, rmses, xlab = "Lambda Penalty Values", ylab = "RMSES")
```



In can clearly be observed that we have accomplished a slight improvement from a non-regularized approach by adding the regularization process through penalizing ratings that have been inflated by insufficient ratings that could have acted as noisy data. As a result, we can identify the penalty λ value that produced the lowest RMSE with the following script.

```
lambdas[which.min(rmses)] # Find the lambda value that produces the lowest RMSE
```

```
## [1] 5
```

It can be determined from both the graph and the script that a penalty λ value of 5.00 produced the lowest RMSE. The lowest RMSE produced is therefore the following, which is a slight improvement from the non-regularized approach.

```r
min(rmses) #lowest value of RMSES
```

```
## [1] 0.8644501
```

4.0 Results and Analysis

We have analyzed using the Loss Model (LM) as a way of predicting movie ratings to be used for movie recommendation. The RMSE is a measure that is used to determine the performance of the derived model, where a lower value indicates a better performance. The following table illustrates the varying LMs used and their corresponding RMSE.

Model	RMSE
Baseline LFM	1.0612018
LFM(M)	0.9439087
LFM(M,U)	0.8653488
LFM(M,U,G)	0.8649469
Regularized LFM(M,U,G)	0.8644501

Where:
`LFM` = Loss Function Model
`LFM(M)` = Loss Function Model with Movie effect
`LFM(M,U)` = Loss Function Model with Movie and User effect
`LFM(M,U,G)` = Loss Function Model with Movie, User, and Genre effect

The baseline model provided us with an RMSE of 1.0612018. As we added more parameters to the baseline model, we managed to achieve better RMSE performances each time. Further performing regularization to the obtained LM as a result of considering the Movie, User, and Genre effect eventually led us to a final RMSE value of 0.8644501.

5.0 Conclusions and Recommendations

We analyzed a subset of the MovieLens 10M Dataset and managed to come up with a prediction model that achieved an RMSE of 0.8644501. As we went on, we started with a baseline model and added different effects found in the dataset that reflected the real life scenario of what determines a movie rating, namely the user preference (user effect), movie preference (movie effect), and genre preference (genre effect) of the general public. Each added effect proved a hypothesis that taking different aspects represented in the dataset could improve the prediction model. To polish the model, a regularization approach was added to add balance to ratings that were a result of a sparse data matrix provided by the dataset resulting to noisy data.

It is understandable that only a subset of the complete MovieLens Dataset was used for this project due to the sheer volume and amount of data. Despite that fact, the dataset still proved to be a challenge for processing data due to hardware and resource limitations. As a result of this, more sophisticated methods such as using Support Vector Machines (SVM), Deep Learning and Neural Networks, and other learning techniques were not used to be able to come up with possibly better results. Likewise, other sophisticated techniques which were hypothesized that could have achieved better results were unfortunately unable to be completed due to the restrictions imposed by the machine where this project was conducted in. Nonetheless, the results obtained from the Loss Models demonstrated here show promise.

Putting hardware limitations aside, recommendations could be to continue using the other aforementioned learning models while using a larger scope of the MovieLens dataset. Likewise, if the MovieLens dataset

can be expanded to include user demographics and actual movie context within the data, this can possibly enable more sophisticated recommendation systems. Given the basis of this project's prediction module to be similar to the grounds of using an approach called Collaborative Filtering, where recommendations are filtered based on the ratings of the general user demographics, it would be a nice recommendation to see if the dataset can be expanded to allow the use of Content-Based filtering techniques in the future.

Bibliography

[1] GroupLens, 2020. *MovieLens Dataset* as published at https://grouplens.org/datasets/movielens/

[2] Irizarry, R. 2020. *Introduction to Data Science* as published at https://rafalab.github.io/dsbook/

[3] Koren, Y. 2009. *The BellKor solution to the Netflix Grand Prize*

[4] Lohr, S. 2009. *Netflix Awards $1 Million Prize and Starts a New Contest* as published at https://bits.blogs.nytimes.com/2009/09/21/netflix-awards-1-million-prize-and-starts-a-new-contest/

[5] Rocca, B. 2009. *Introduction to recommender systems* as published at https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada