MovieLens Prediction Model

by Umer Abid Ali

Overview

The following is an account of my attempt at building a Machine Learning model to predict the ratings given by users to a set of movies present in the **MovieLens** dataset, a public dataset that contains information of the ratings given to different movies by a whole host of users. The objective of this project was to design a prediction algorithm that predicts user ratings for different movies with an estimated *Root-Mean-Square error* of 0.8649 or less. The following is a report on my experience of building said algorithm and a statistical overview of its performance.

The Dataset

As mentioned before, the dataset used for building this project is the publicly available MovieLens dataset. The dataset contains approximately 10 million ratings given to a total of 10,000 movies by 72,000 users. Each rating (row in the data table) has the following fields of data associated to it:

- **User ID** - The unique ID of the user who's given the rating (referred to as *userId* in code)
- **Movie ID** - The unique ID of the movie that was rated (referred to as *movieId* in code)
- **Rating** - The rating given to the movie by the user. This is a number between 0.5 (lowest/disliked by the user) and 5 (highest/liked by the user) and is referred to as *rating* in code
- **Title** - This is the title of the movie being rated (referred to as *title* in code)
- **Genres** - This is a pipe (|) separated list of all the genres the movie falls into (referred to as *genres* in code)
- **Timestamp** - This field contains information regarding when the review was registered as it stores the number of seconds that have passed between the *epoch* date, 1970-01-01, and the time of registration of the review (referred to as *timestamp* in code)

The original dataset was divided into two subsets via the project distribution code provided by Edx. The names and purposes carried by these subsets are described below:

- **edx** - This dataset was to be used to train the prediction model and constituted of approximately 90% of the original dataset
- **validation** - This dataset was to be used to test the final prediction model and is also referred to as the *final hold-out test set* and constituted of the remaining 10% of the original dataset that was not present in the edx dataset

A glimpse of the data stored in the above described datasets is given below:

```
#Displaying the first 3 ratings in each data set
edx %>% head(3)
```

```
##    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)
##                           genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3: Action|Drama|Sci-Fi|Thriller
```

```
validation %>% head(3)
```

```
##    userId movieId rating timestamp                title
## 1:      1     231      5 838983392 Dumb & Dumber (1994)
## 2:      1     480      5 838983653 Jurassic Park (1993)
## 3:      1     586      5 838984068    Home Alone (1990)
##                               genres
## 1:                            Comedy
## 2: Action|Adventure|Sci-Fi|Thriller
## 3:                   Children|Comedy
```

Steps and stages involved

There were 4 main stages that were involved in the building of the prediction model. These are described below:

1. **Data Preparation** - This stage involved the extraction, division and cleaning of the MovieLens data
2. **Data Exploration** - This stage involved exploring the data and gaining insights in order to determine which features of the data affected the outcome (rating) the most
3. **Building the model** - This stage involved incorporating the results of the data exploration done in the previous stage and combining them into a prediction model
4. **Training the model** - This stage involved training the model on a test set in order to determine whether or not it produces expected results

All 4 stages defined above are described in detail over the course of the following two sections, namely **Data Analysis** and **Model Performance**

Data Analysis

Discussed here are the different methods of analysis and exploration employed in order to gain insights into the data and discover where the variability in the ratings originates from.

Data Preparation

Majority of the necessary extraction and cleaning of the original data is handled by the distribution code provided by Edx. However, as per the instructions for the project, the validation data set was to be used **only** on the final, fully-prepared prediction model and therefore could not be treated as a test set when building the model.

This void of a readily available test set meant that the *edx* data set, which was provided for training, had to be split into 2 further subsets, one for training and one for testing. In order to do this, I used the same configuration as the distribution code and split the *edx* data set in a 90-10 manner where 90% of the data became the training set (referred to as *train* in code) and 10% became the test set (referred to as *test* in code). Of course these ratios weren't preserved perfectly as the test set had to be filtered further in order to ensure it only contained ratings of movies that were present in the training set and that those ratings were given only by users whose data was present in the training set. A series of *join* functions were used to achieve this in a much similar manner to the distribution code. Below is the code for the generation of an *index* by which the edx data set was split.

```r
set.seed(1, sample.kind ="Rounding")
#Creating an index that will subset 10% of the edx data, p = 0.1
index <- createDataPartition(y = edx$rating, times = 1, p = 0.10, list = FALSE)
#Defining the training set as a subset of edx that is not referred to by index (90%)
train <- edx %>% slice(-index)
#Defining the test set as a subset of edx indexed using the above created index
temp_test <- edx %>% slice(index)
```

Notice that in the above code the test set is referred to as *temp_test* and not *test* as it still needs further tuning in order to be fully prepped.

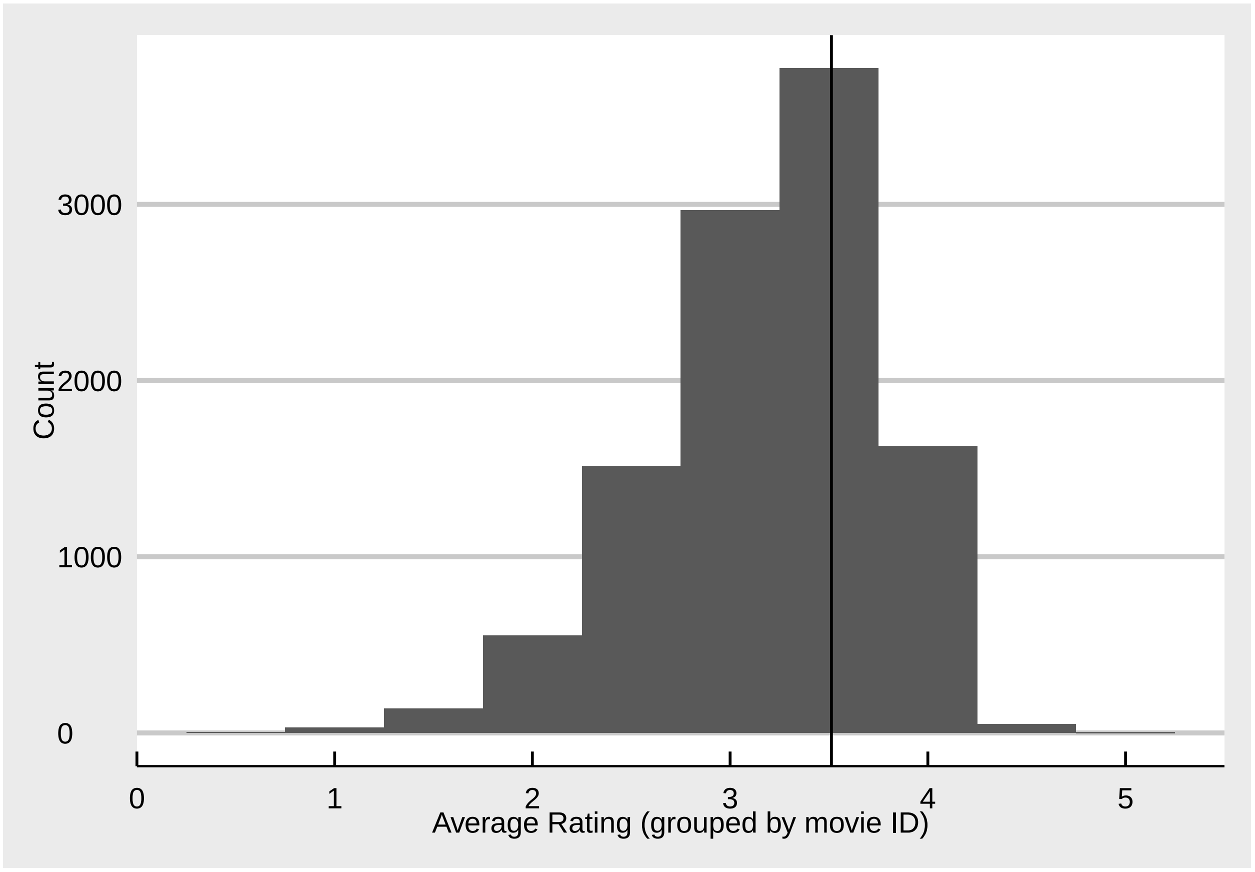
Data Exploration

There are many different features of the data that may carry an effect on the final rating given to a movie by a user. After a first look at the data, some of these are immediately visible, namely:

- **Movie Effect** - The quality and scope (viewership range) of the movie being reviewed. There are many sub factors that come into play when it comes to the quality and scope of a movie, some of which may be:
  - The studio that produced the movie
  - The budget that the studio invested in the movie
  - The cast of the movie
    However, as these data are not available in the data set, their effects will be encapsulated in a net movie effect
- **Genre Effect** - The genre(s) that the movie falls into may have an effect on the movie's average rating, for example, widely-watched genres such as Action or Comedy are more critiqued than the more scarce ones such as Documentary or Biography and thus have lower average ratings. However, this perception cannot be assumed to be true until it has been proved by data exploration
- **User Effect** - The critique-ness of a user i.e. how easily is the user impressed? (An easy going user will give a higher rating to all reviewed movies on average and vice versa)
- **Time Effect** - The time that a movie was reviewed may matter as well, for example, a younger user may not be fond of 19th century movies and vice versa. However, this statement too needs to be put to the test via data exploration

Movie Effect

I first dived into the effects a movie's quality and scope carries on its rating. To start off, I made a histogram of the average ratings given to all movies in the training set. Below is a histogram of the results with the vertical line representing the average rating across all the movies in the training set:



As can be seen from the histogram, the ratings are approximately normally distributed centering around the average rating for all movies which is 3.5125. However, it must also be noted that a significant number of movies have ratings that are not close to the average rating. This claim can be further cemented by looking at the standard deviation of the ratings:

```r
sd(train$rating)
```

```
## [1] 1.06
```

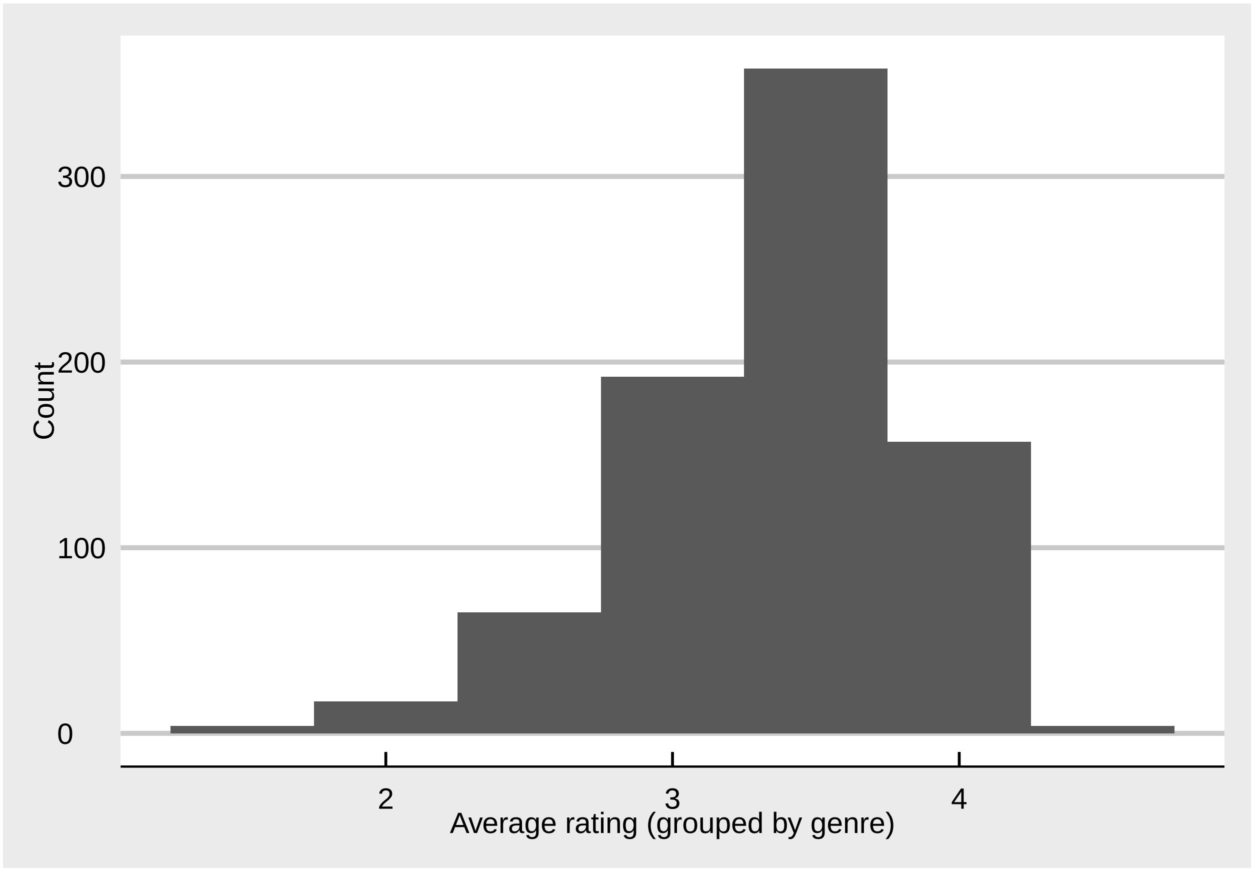
Therefore, taking the average rating for a movie to make a prediction seems to be a good first piece of the puzzle. Hence, I decided to create a data frame that contained the average ratings for all movies along with their unique IDs, the number of times they had been rated, and their respective genres (reasons explained later). Below is the code used to do so:

```r
#Computing the average rating given to each individual movie
#And the number of times it was rated
movie_avgs <- train %>% group_by(movieId) %>% summarize(avg = mean(rating),
                                                        movie_n = n(),
                                                        genres = genres[1])
```

Genre Effect

When creating my data frame containing average movie ratings, I came to a valuable insight that there are many movies that have been very rarely rated and therefore their average rating may not be an as useful metric to use when making predictions. Regularization didn't seem to be a plausible solution in this case therefore I decided to go with the next best option which was to collect more data about those *scarce* movies by looking at other similar movies i.e. other movies with the same genres.

I proceeded to create a similar data frame to the one that housed the average movie ratings only this one composed of the average ratings carried by each respective genre. I then generated a histogram of these ratings and saw a similar distribution to that of the average movie ratings.



Hence, I came to the conclusion that I could use a combination of the average ratings across all movies and the average ratings across all genres to make predictions.

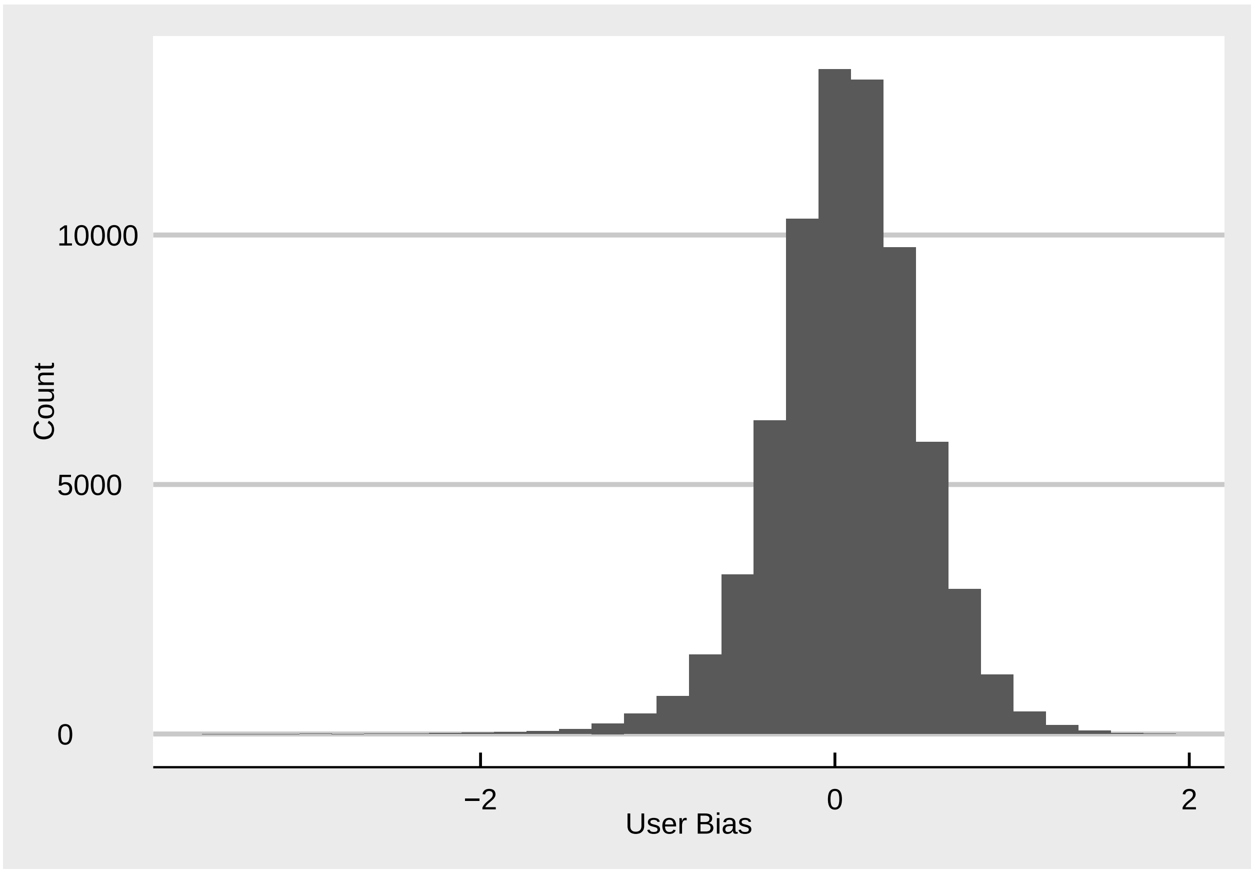
I decided to define a cutoff for the number of times a movie was rated which would classify whether a movie is *scarce* or not. This cutoff was set at **15** ratings i.e. a movie must be rated at least 15 times in order to classify as *non-scarce* hence leading to the creation of a new data frame that contained the average ratings for scarce movies only (referred to as *movie_avgs_scarce* in code). The averages of these movies would then be redefined as the average ratings of their respective genres whilst the average ratings of the non-scarce movies would remain the same. Below is the code used to create the final data frame containing average ratings for both scarce and non-scarce movies:

```
#Filtering out non-scarce movies
movie_avgs_final <- movie_avgs %>% filter(movie_n > 15) %>% select(movieId, avg)
#Combining scarce and non-scarce movies' data together
movie_avgs_final <- movie_avgs_final %>% rbind(movie_avgs_scarce)
```

User Effect

Now that the movie and genre effects were out of the way, the focus shifted to the next presumed significant factor in deciding the rating given to a movie by a user - the user themself. I decided to follow a different approach when conducting exploratory data analysis for user effects; rather than visualizing the average *rating* given by each user to all movies they have rated, I decided to visualize the average *bias* the user tends to have when giving a rating. The reason behind this lies in the human nature of each user i.e. each user naturally has a preference towards a certain genre or movie therefore might rate that movie or movies of that genre higher on average than they would rate other movies or movies of other genres, the subject (the movie being rated) is not constant for each user like it was in the case of movie effects thus making calculating biases a better option.

The bias each user carries towards a rating is computed by removing the already explained variability (movie effect) from a rating and determining the residual which is presumed to be due to the user's bias i.e. it is the mean of the residual obtained after deducting the movie and/or genre effect from a rating. The following histogram visualizes the distribution of this value (referred to as *B_u* in code):

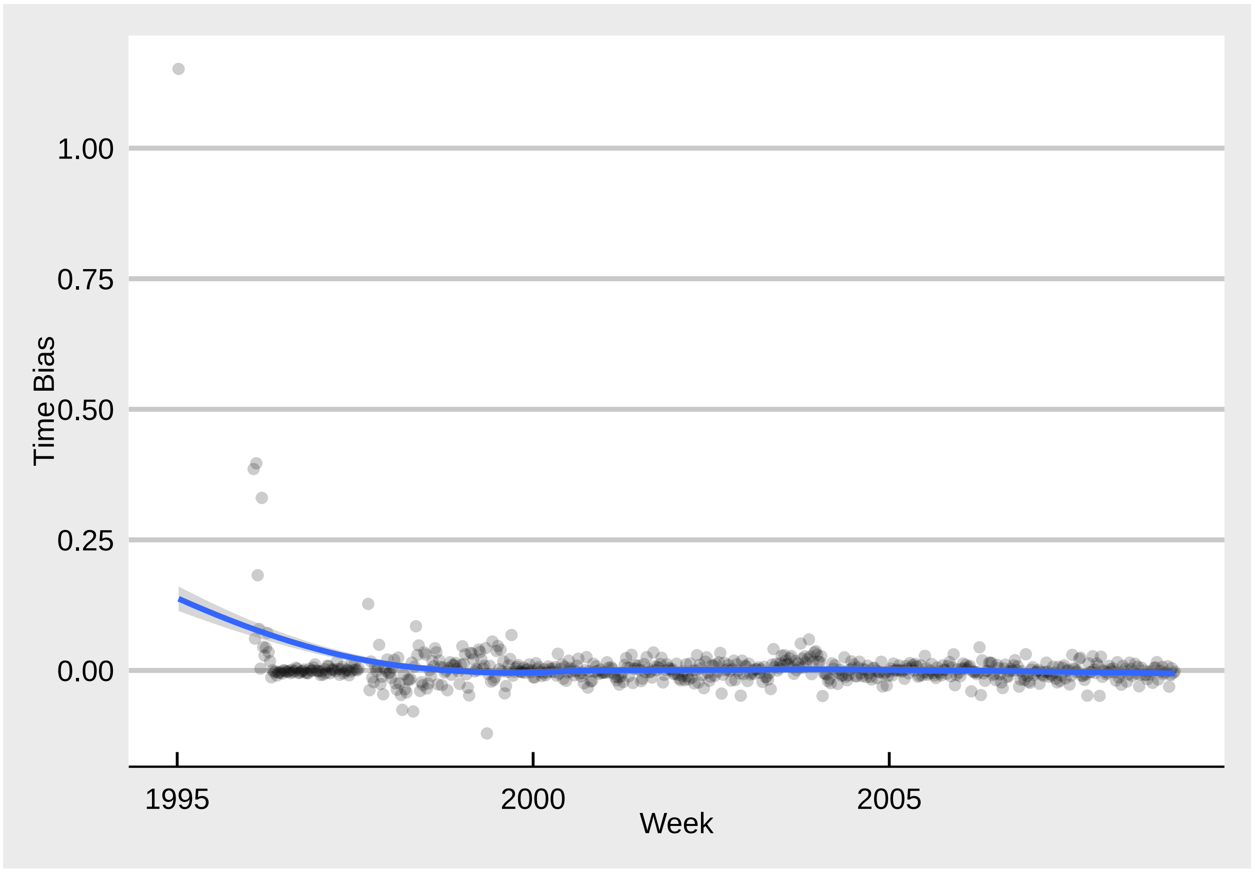


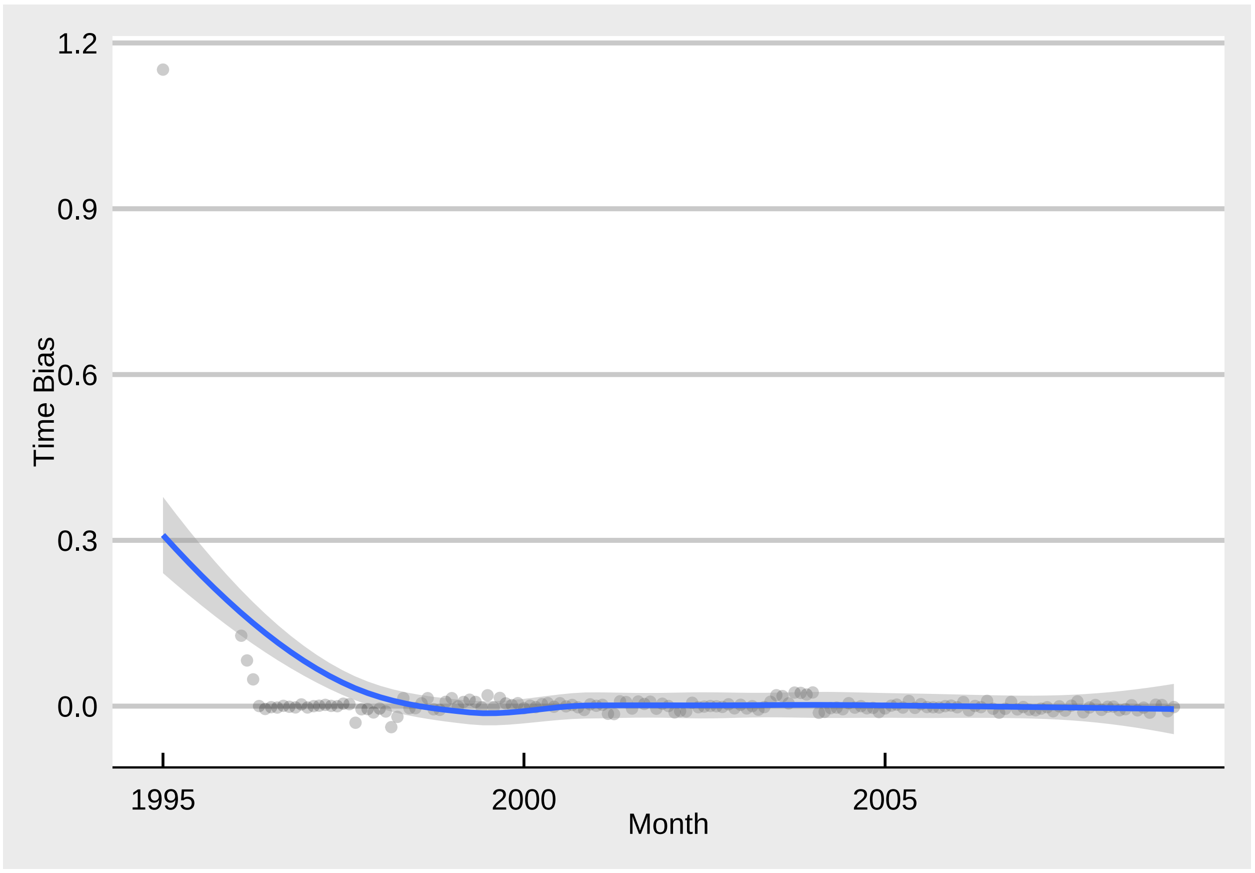
As can be seen from the histogram, although the distribution seems to be centered around 0 and is approximately normally distributed, a significant portion of the distribution is covered within -1 $< B_u <$ 1 thereby serving as evidence of a user bias.

Time Effect

The time at which a review was registered may carry an effect on the rating given by the user in the review. As alluded to earlier, an example of such a case would be the fact that a younger user may not be fond of 19th century movies and may give low ratings to the otherwise popular and highly rated movies such as *The Godfather* or *The Goodfellas*.

To conduct exploratory data analysis on this analogy, I employed a similar approach to that of computing the user effects (see above) and calculated a net *bias* for the time at which a review was registered. I visualized this bias when calculated at two different intervals of time, on a *weekly* basis and a *monthly* basis. Below are scatter plots for both computations with *weekly* biases shown first. A smooth line of best fit is also shown crossing through the points.





As can be seen from the above plots, there is little to no variability between the biases both when computed on a weekly and monthly basis therefore rejecting the hypothesis that the time of registering a review has a net effect on its rating.

Model Performance

Now that exploratory data analysis was complete and important insights on the data were gained, it was time to start building the model.

Building the Model

I built a primary model which focused mainly on the two big presumed sources of variability in the ratings - The **movie** and **user** effect. The model was built upon two main formulas out of which one was used to make a prediction based upon the classification of a movie as scarce or not ($n < 15$). The first of the two equations used is given below:

$$Y = \mu_M + B_u$$

Where the definition of each symbol is as follows:

- Y - The predicted rating
- μ_M - The mean rating of the movie in question
- B_u - The net bias associated with the user in question (user bias)

This equation was used only on the restriction that the movie in question was not classified as *scarce* ($n > 15$)

The other equation was reserved only for movies classified as *scarce* ($n < 15$) and was as follows:

$$Y = \mu_G + B_u$$

Where the definition of each symbol is as follows:

- Y - The predicted rating
- μ_G - The mean rating across all movies having the same genre as the movie in question
- B_u - The net bias associated with the user in question (user bias)

However, as we had already joined the average movie and genre ratings into a single data frame in accordance to the classification of each movie, the two equations mentioned above were merged into one in the final model:

$$Y = \mu + B_u$$

Where the definition of each symbol is as follows:

- Y - The predicted rating
- μ - The mean rating given to the movie or the mean rating across all movies having the same genre as the movie in question, based upon the classification of the movie
- B_u - The net bias associated with the user in question (user bias)

Training the Model

With the prediction algorithm seemingly fully designed, it had to be put to the test. However, before doing so, a metric had to be decided in order to quantify the performance of the algorithm - this metric was the **Root-Mean-Square Error (RMSE)** as directed by the project objectives. Hence, a loss function had to be defined to compute the estimated RMSE in the predictions made by the model. The following code was used to define said function:

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

Now we were ready to start making predictions; a vector containing all the predicted ratings for each review in the test set was created using a series of *join* functions in order incorporate the movie/genre average ratings (μ) and the net user bias (B_u) followed by the employment of the above derived model equation. The vector was named *preds* and yielded the following RMSE.

```
RMSE(test$rating, preds)
```

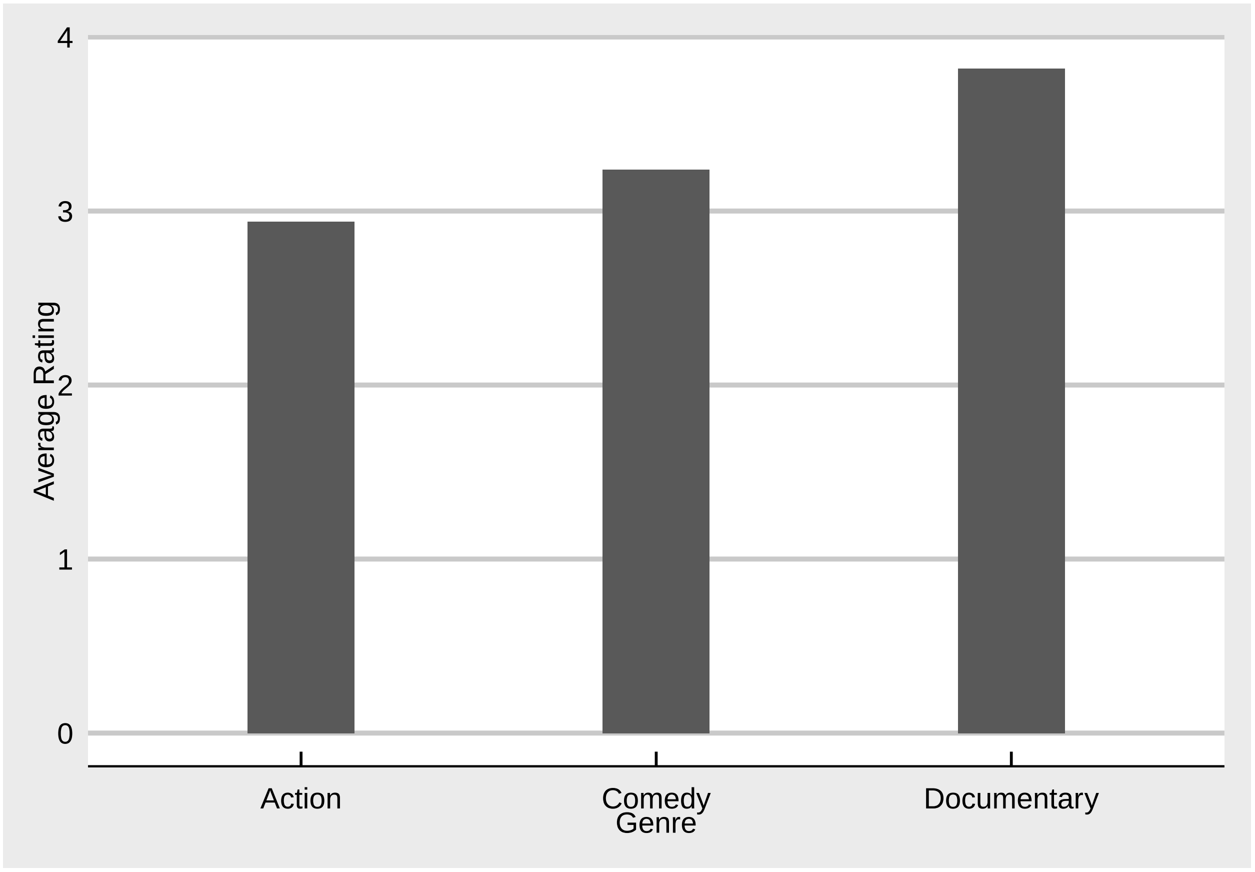
```
## [1] 0.865
```

As can be seen, though the computed RMSE is around the required 0.8649 objective, it still needs further curtailing in order to be acceptable. This means that there's still some variability in the ratings that we haven't been able to explain. Therefore, the results had to be re-assessed in order to check which predicted ratings were the furthest from their actual values. Below is a list of the top 5 movies with the highest residuals (difference between prediction and actual rating):

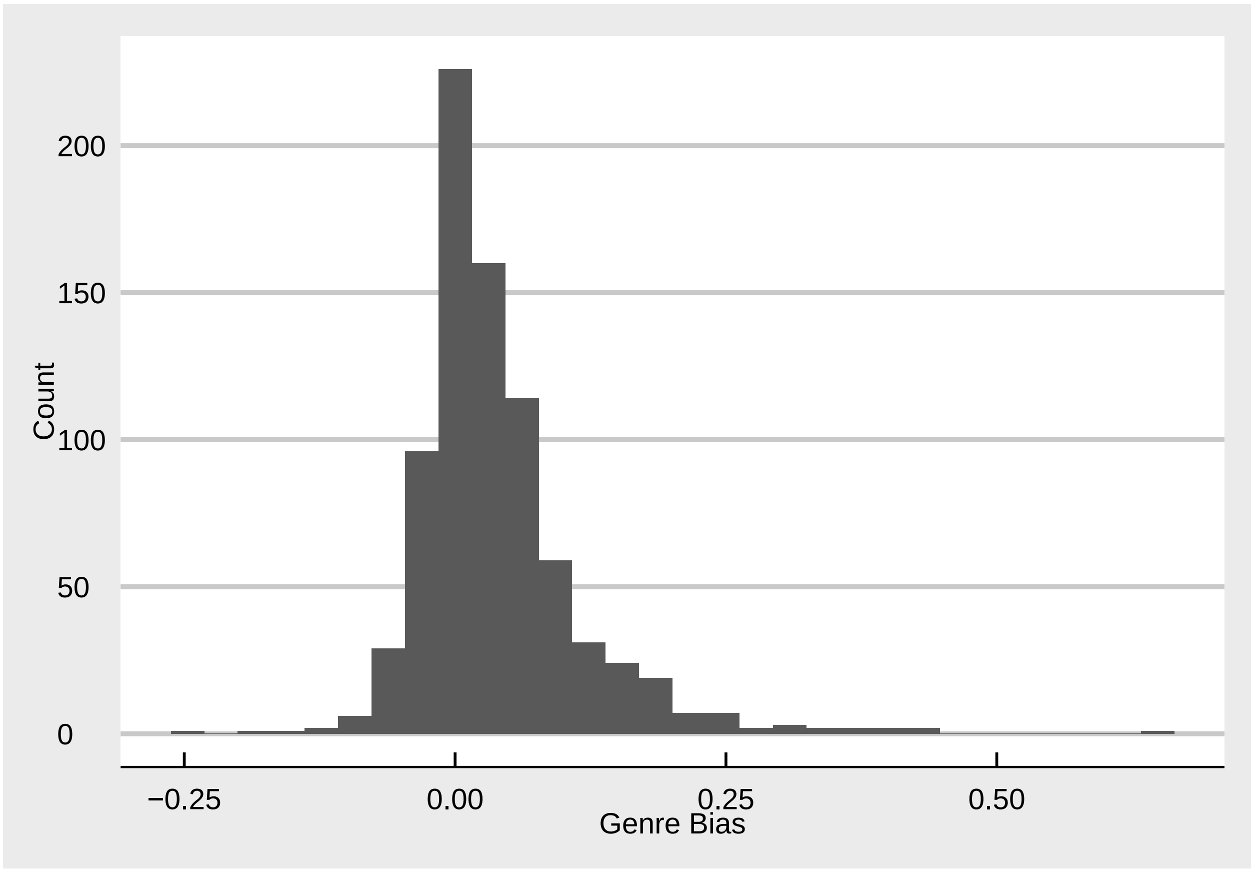
```
##    movieId userId rating    pred residual    avg    B_u
## 1:    6587  21267      5 -0.2359    5.236 1.1975 -1.433
## 2:    5248  10803      5  0.4303    4.570 1.8584 -1.428
## 3:    6483  29924      5  0.6103    4.390 0.8743 -0.264
## 4:    1930  26150      5  0.6559    4.344 2.9074 -2.251
## 5:    2541  19059      5  0.6672    4.333 3.2672 -2.600
```

As can be seen, although some of the average movie ratings are close to overall average of 3.5125, they are annulled by the negative user biases. However, the actual ratings do not suggest this to be true.

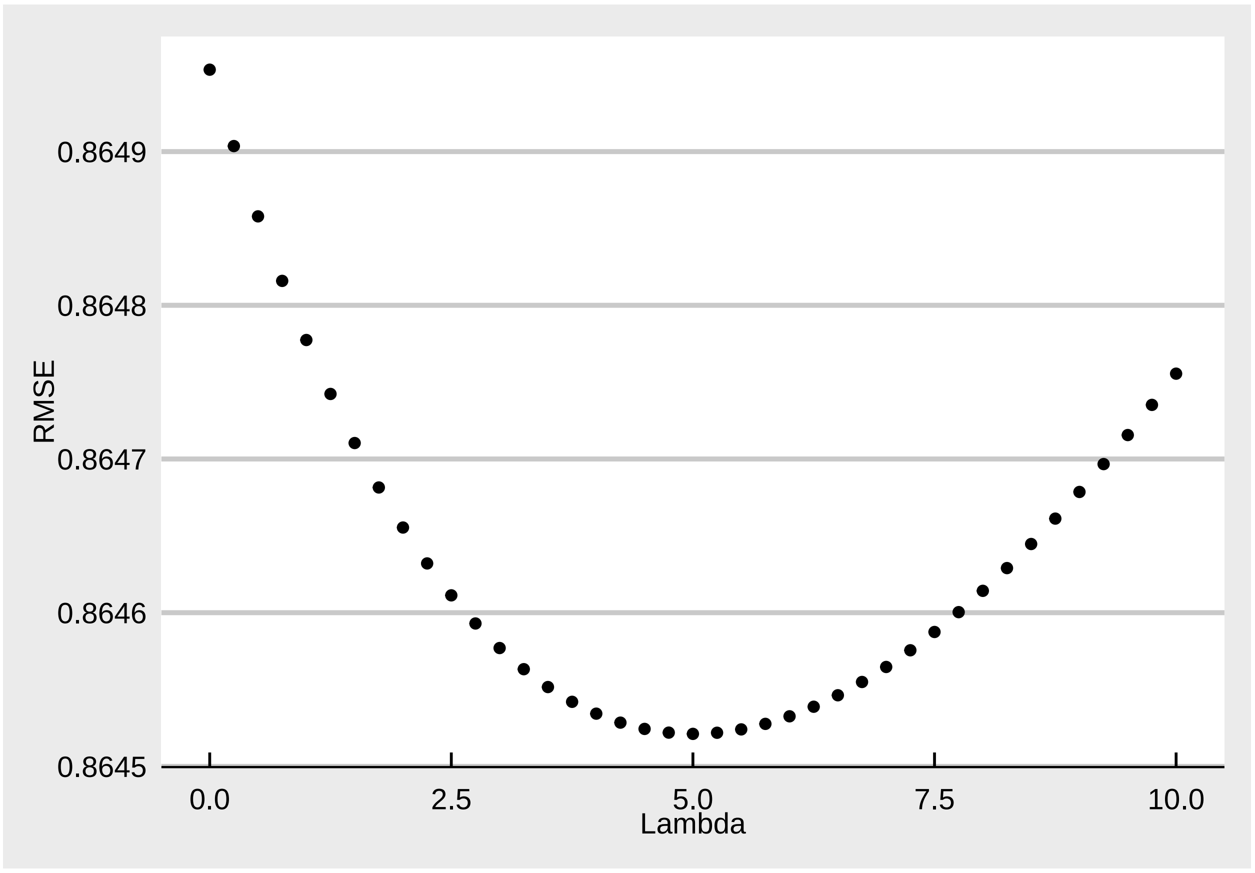
Since we have already determined that the time of registration of the review doesn't have any effect on the rating it has, it's safe to assume that some of this unexplained variability originates from the genre of the movie. Recall that, although we have taken the average rating across all movies of a genre into account, we have done so only for the genres whose movies are classified as *scarce*. There has been no consideration of genre in the case of non-scarce movies. Therefore, a net *bias* has to be computed that is associated with a movie's genre. The bias seemingly originates from the presumption mentioned earlier, that widely viewed and blockbuster genres such as *Action* or *Comedy* are more frequently critiqued and therefore have a lower average rating when compared to the more scarcely critiqued ones like *Documentary* or *Biography*. Exploratory data analysis also backs this presumption as can be seen in this bar plot below:



Looking at the chart on the previous page, its safe to say that genre does carry an effect on the rating given to a movie which can be quantified again using a similar approach to the one used when calculating user effects in which the pre-explained variability (movie and user effects) is removed and the mean of the residual is considered to be the net bias due to the feature in question (genres). Below is a histogram showing the distribution of these computed biases (note that genre biases for movies classified as scarce will be set to 0 when running the model as the average ratings of these movies is already derived from their respective genres).



Furthermore, the user biases seem to be inexplicably high/low than they are expected to be. This might be because of the fact that not all users rate the same amount of movies and therefore biases for users with few reviews have high variability. Regularization may be a solution to this problem as it will shrink the biases for users who have few reviews. However, it first needs to be determined what the value of the regularization parameter λ should be such that it minimizes the RMSE. Cross-validation was employed in order to do so. Below is a scatter plot of a set of values ranging from 0 to 10 with intervals of 0.25 between them which were tested as λ and their respective RMSEs:



As is visible, the RMSE minimizes at $\lambda = 5$ where it is about *0.8645*. Therefore, the user and genre biases were regularized with $\lambda = 5$ and re-computed.

Another factor that may be unnecessarily increasing the RMSE is the range of the predictions. As was visible in the above list of top 5 movies with the highest residuals in their predicted ratings, some of the predicted ratings were below 0 which, obviously, is not possible. Therefore, there is a need to standardize these predictions and make them all be within the range of $0.5 \leq Y \leq 5$. To do so, a function was defined which checked whether or not each prediction was within plausible boundaries and if not, corrected them to be within said boundaries. Below is the code used to define the function:

```
standardize_preds <- function(pred){
  #if prediction is lower than lowest possible rating
  #Then redefine it to be the lowest possible, 0.5.
  if (pred < 0.5){pred <- 0.5}
  #if prediction is higher than the highest possible rating
  #Then redefine it to be the highest possible, 5.
  else if (pred > 5){pred <- 5}
  #if prediction is within the range of actual predictions
  #Then leave it as it is.
  else {pred <- pred}
  return (pred)
}
```

Results

With the final tweaks and adjustments having been made, the model was now complete and the objective RMSE of 0.8649 or less was achieved as well. All that was left was to test the model on the *final hold-out test set*, the **validation** set, after training it with the entirety of the **edx** data. The following is a symbolic representation of the final model in the form of a mathematical equation:

$$Y = \mu + B_u + B_g$$

Where the definition of each symbol is as follows:

- Y - The predicted rating
- μ - The mean rating given to the movie or to all movies having the same genre as the movie in question, based upon the classification of the movie in question
- B_u - The net bias associated with the user in question (user bias), regularized with $\lambda = 5$
- B_g - The net bias associated with the genre of the movie in question (genre bias), regularized with $\lambda = 5$

Below is the computed RMSE obtained when training the model using the entire edx data set and testing on the final validation set followed by standardization of predictions:

Final RMSE computed: 0.8646

Conclusion

In conclusion, the model successfully managed to shrink the RMSE to the required 0.8649 or less with the help of incorporating the movie, user and genre effects and a couple of optimization techniques such as regularization and standardization. However, it must come as no surprise that it is somewhat complicated making it difficult to implement due to its high computing power demands. That is one of the major areas of this model that require improvement - the efficiency of the model in terms of processing power and RAM consumption. Overall, however, it performs as expected and produces the required results thus making it, in my eyes, a success!