MovieLens Report

Md Ishtiaque Hossain

8/21/2019

1. Executive Summary

The dataset being used for this project is the 10M version of MovieLens dataset compiled by GroupLens research. The key features of this dataset are userId, movieId, genre, timestamp, title and lastly, the feature that we are trying to predict; rating.

The goal of this project is to create a rating prediction model using the train set, which can be used to predict ratings for movies in the validation set. The predicted ratings will be compared to true ratings in the validation set using RMSE (Root Mean Square Error).

According to project guidelines, to score full 25 points, RMSE needs to be lower or equal to 0.8649. In order to accomplish this, a predictive model was created using training data, by taking user bias, movie bias and genre bias into account. Then regularization was used to further reduce RMSE.

DataSet

For this project a movie rating predictor is created using the 'MovieLens' dataset. This data set can be found and downloaded here:

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

Data Loading and Setup

We use the following code to create training and validation sets.

Create edx set, validation set

Note: this process could take a couple of minutes

2. Methods and Analysis

Methods Used

We used the following model to predict movie rating:

Y = mu + b_i + b_u + b_g + i_e

Y is our prediction, mu is the average, b_i is movie bias, b_u is user bias, b_g is genre bias and "i_e" is the independent error in our predictions. We also use regularization to penalize large estimates formed using small sample sizes.

Exploratory Analysis

A quick review using the head() function shows that there are six columns in total. userId and movieId are unique identifiers for users and movies respectively. Rating column contains the ratings given by individual users. Timestamp records the actual time the rating was given. Title contains the name of the movie and the year it was released. Genre corresponds the genre of the movie. The genre column is rather interesting and it is easy to see that each movie can be simultaneously part of multiple genres.

```
##   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
```

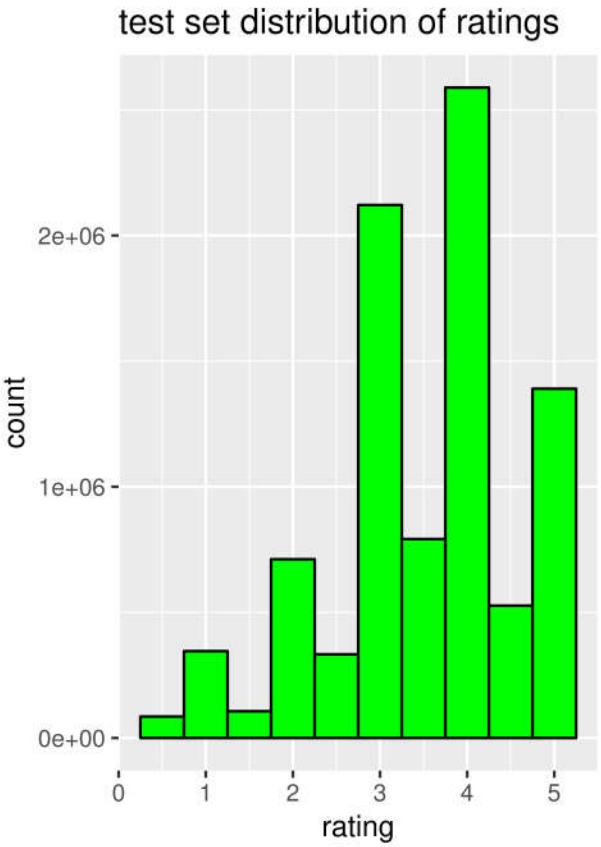
We can take a quick look at the distinct number of users and movies in our training set:

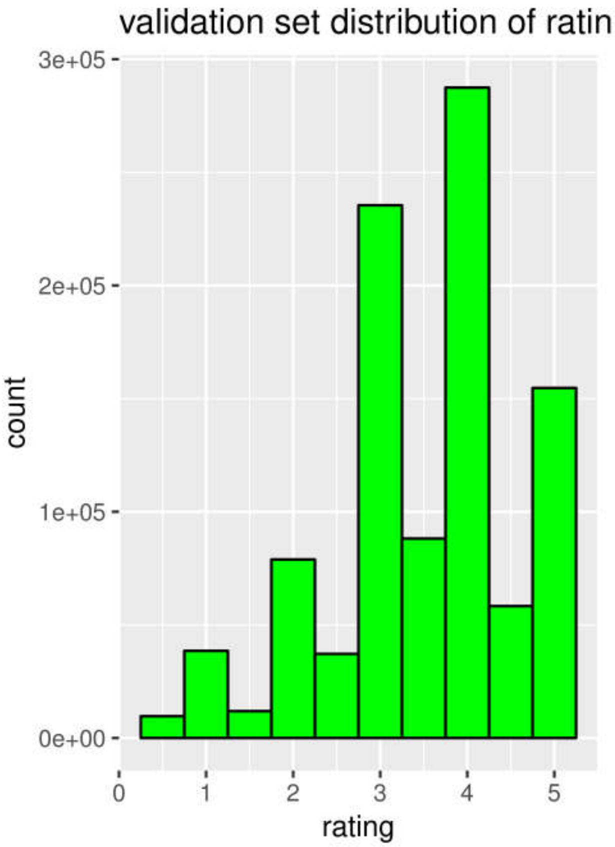
```
##   n_users n_movies
## 1   69878    10677
```

And also the distinct number of users and movies in the validation set:

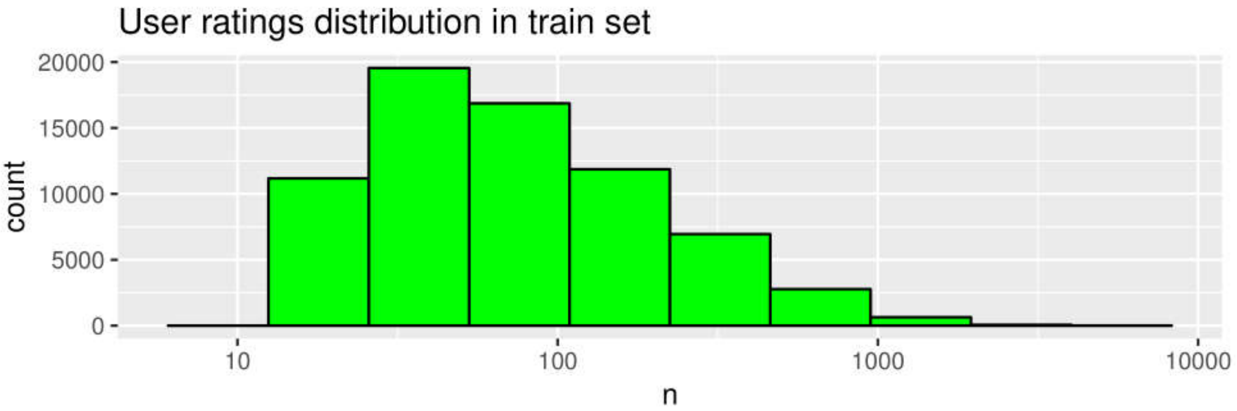
```
##   n_users n_movies
## 1   68534     9809
```

We see that the number of movies and users in both sets are roughly the same. Now let's look at the ratings distribution in the test set:

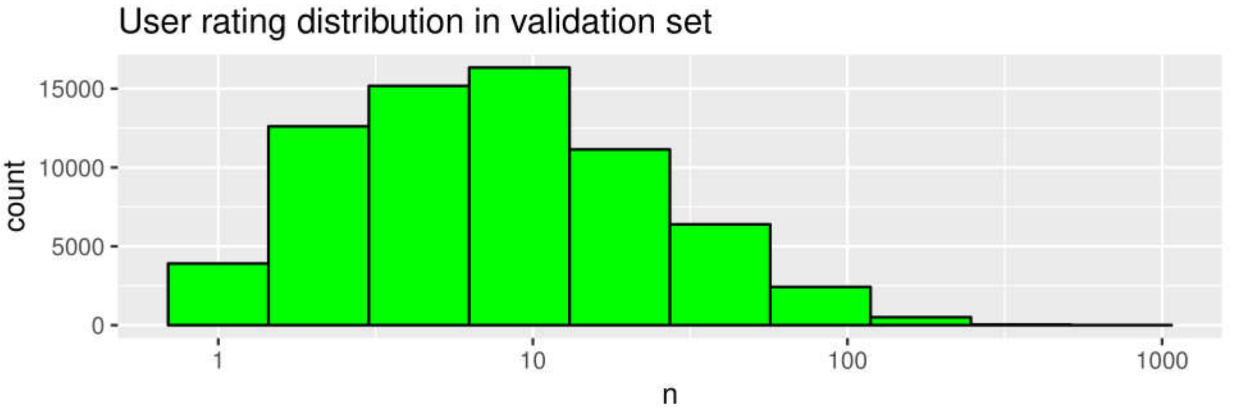




From the aforementioned charts, we can see that the distribution of ratings in both sets are roughly the same as well. Now let's look at how active/prolific the users are in rating movies in each dataset.



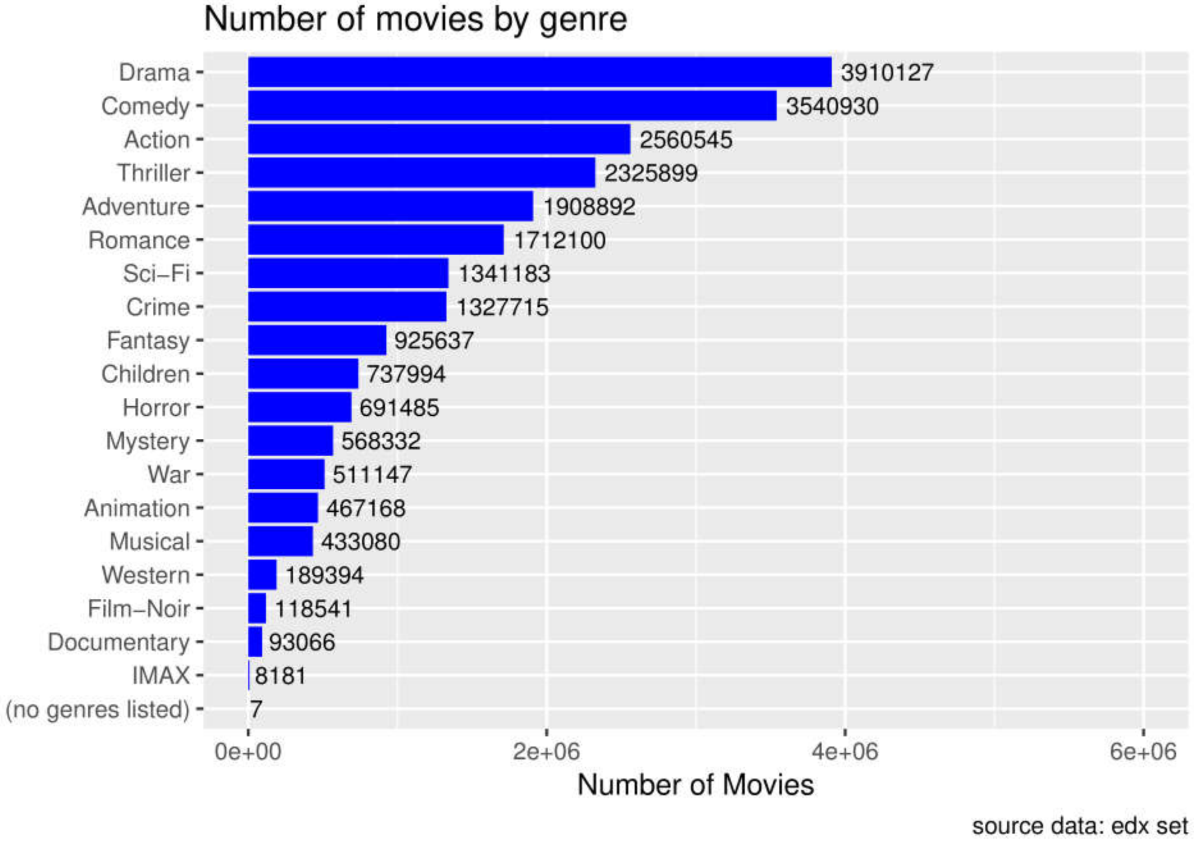
User ratings distribution in train set



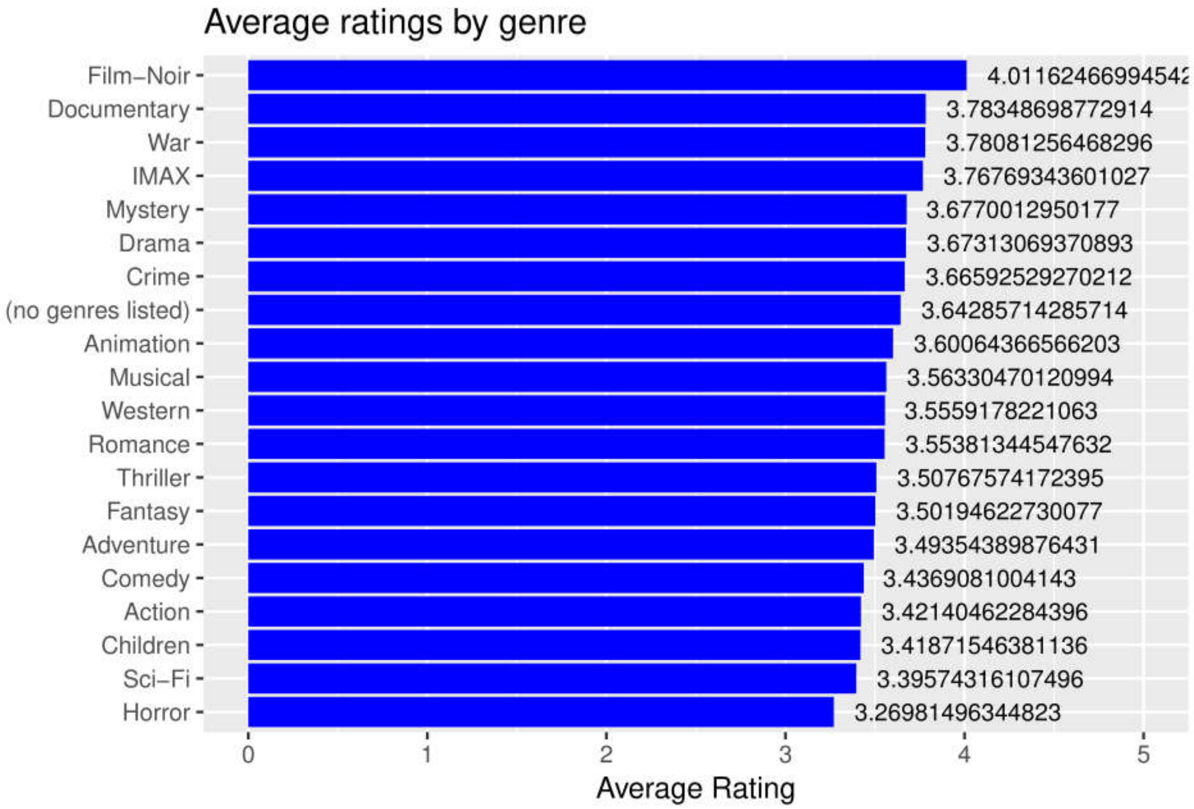
User rating distribution in validation set

We can see some users are more active in rating movies than the others. We should note that the train set has more prolific/active raters (users who have rated more than a 100 movies) compared to users in the validation set.

We initially noted that in the genre column, each movie can contain multiple genre. In this form, it is hard to see how many movies are in each individual genre. Therefore we will split the genre row so each movie gets a single genre value in each cell. Movies that belong to multiple genre will have multime observations because we are going to split each genre into it's own row. This makes it easier to calculate the number of movies in each genre. We can see that drama has the highest number of movies, and documentary, IMAX and "no genres listed" having the lowest number of movies.



Now we look at the average ratings in each genre.Turns out Film-Noir and Documentary are the highest rated movies on average, while Sci-Fi and Horror are the lowest rated ones.



source data: edx set

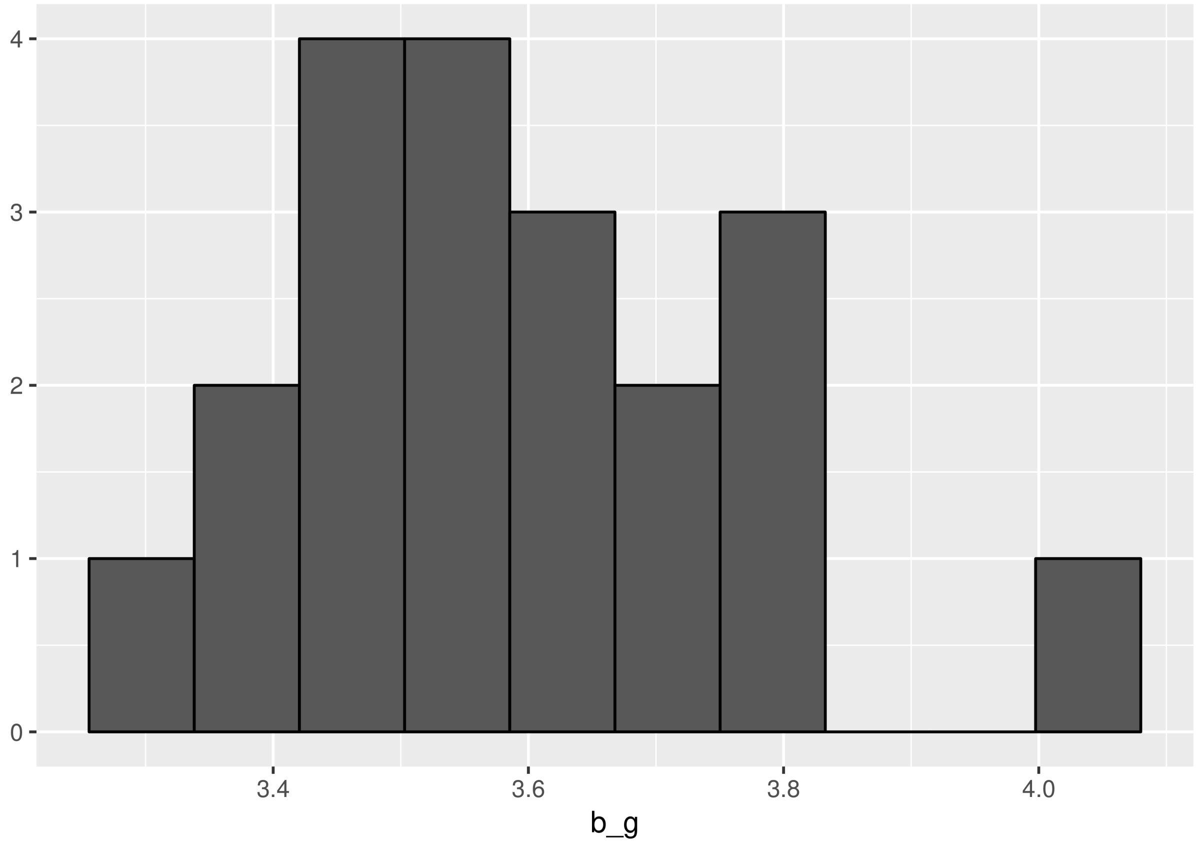
Now let's look at the overall average in the edx set.

```
# Calculating overall average rating across all genres

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

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

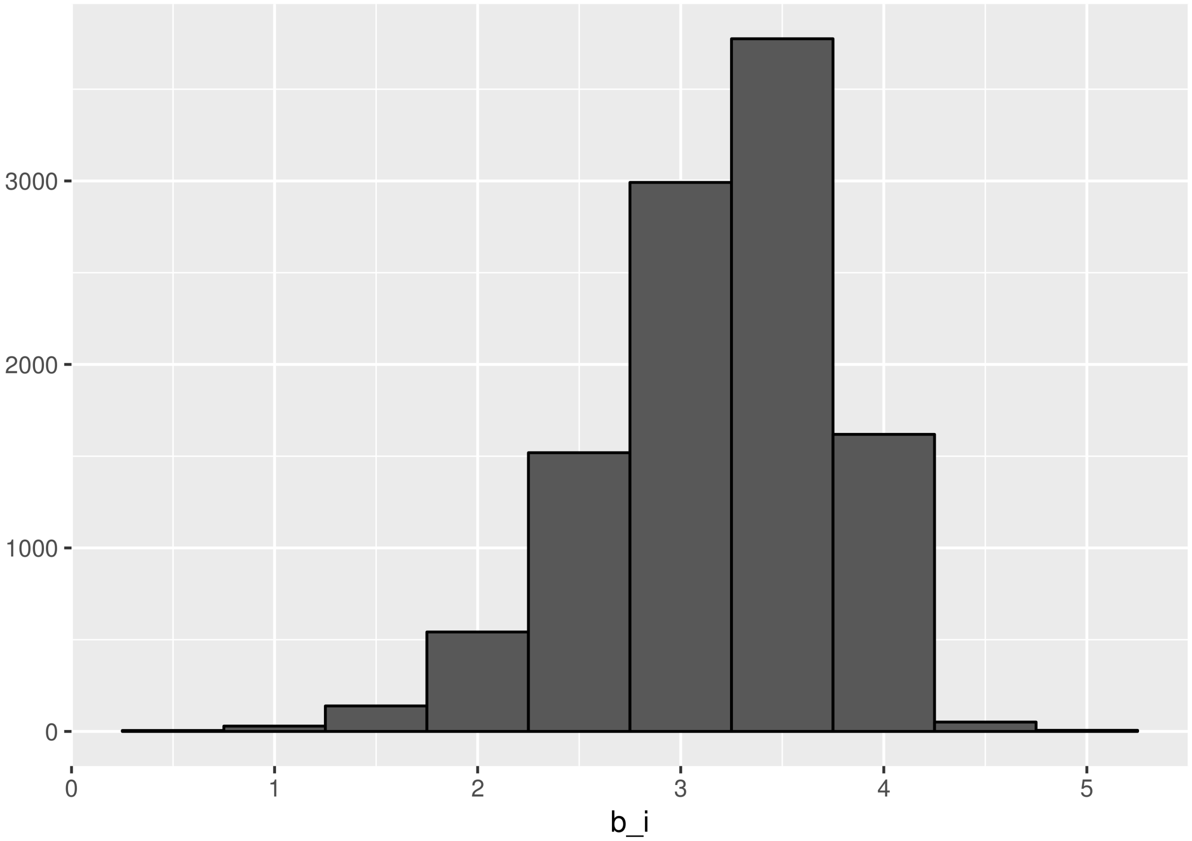
As we can see the average rating across all genres is 3.51, so we can assume users are rather generous with their ratings, since it's more than 70% across all movies.

Visualizing genre effect



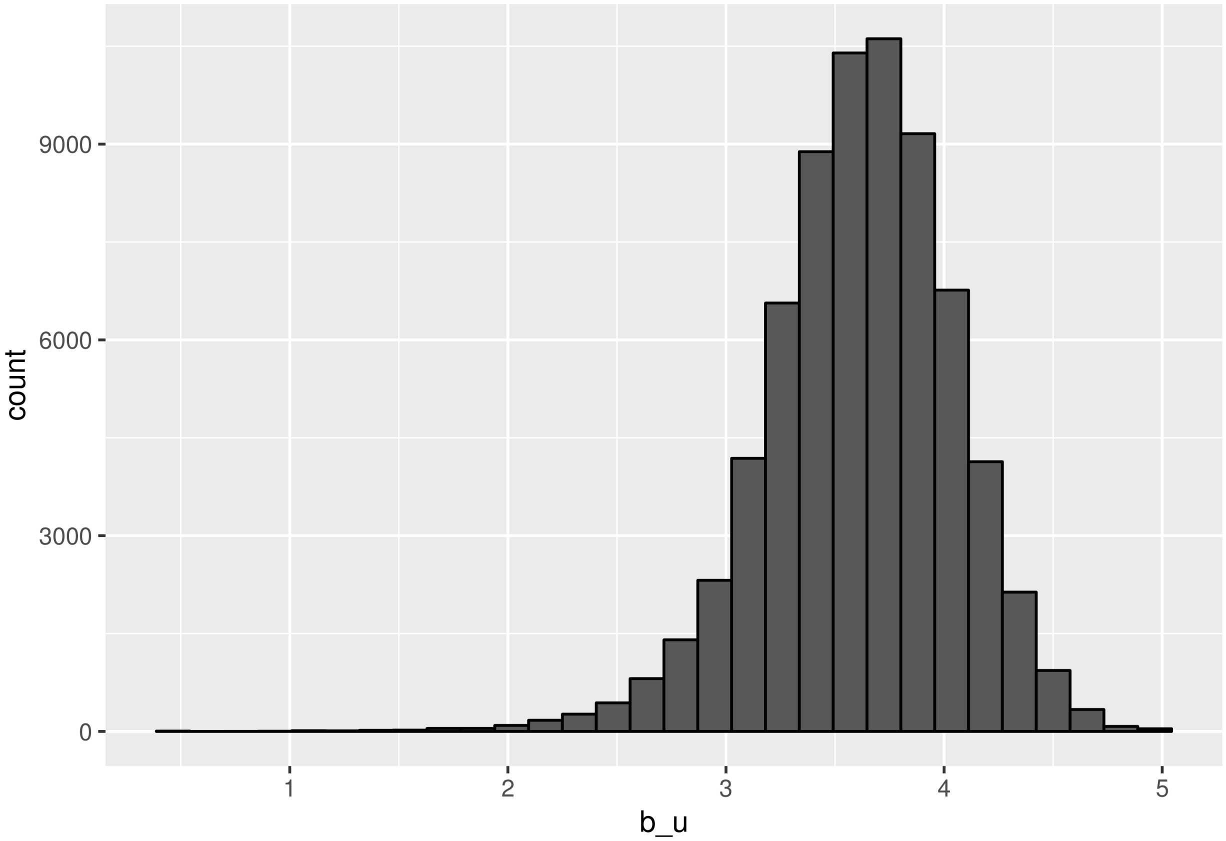
But once we group ratings by genre and visualize it using a histogram, we see that there is high variability in rating averages from genre to genre. Therefore it is safe to declare that the genre of the movie has a significant enough effect on its rating for us to consider it for modelling.

Visualizing movie effect



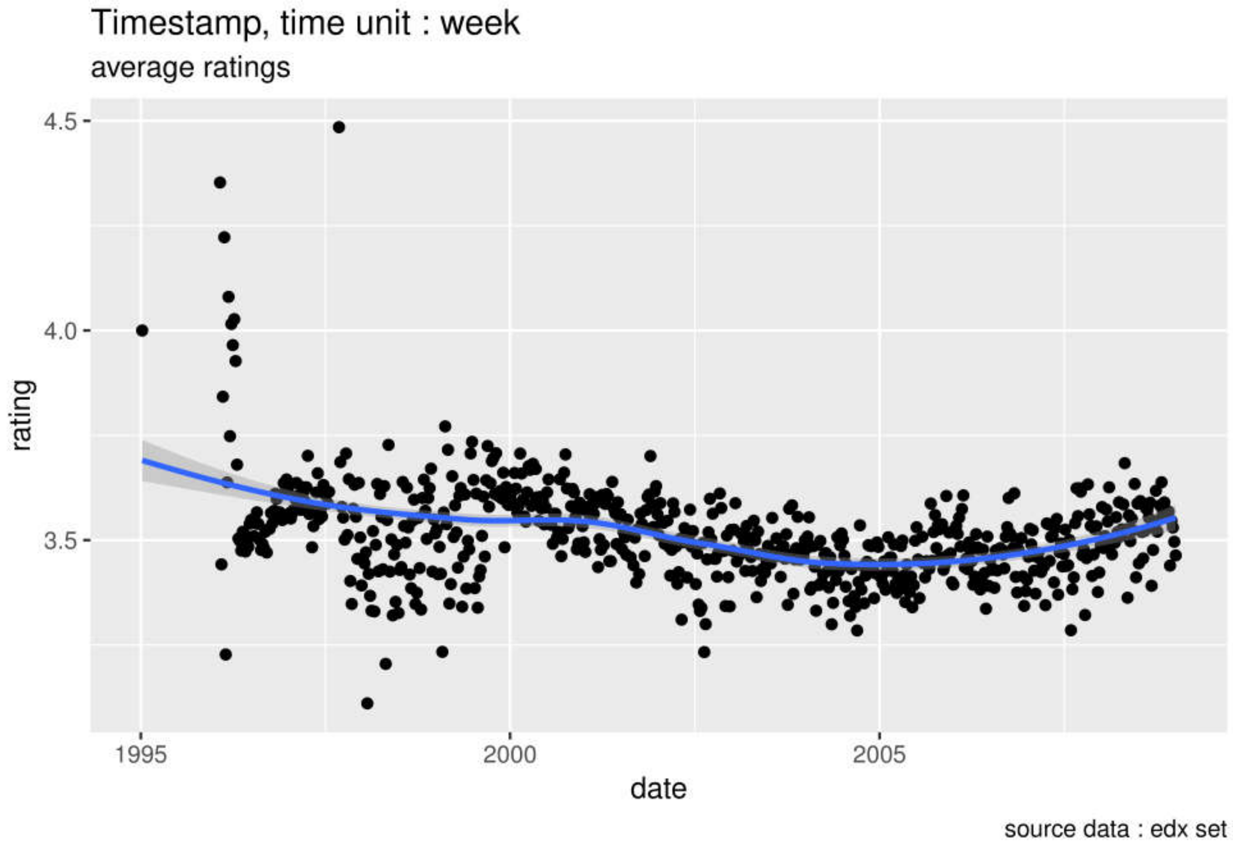
Once again, we observe high variability in ratings depending on the movie. Which, intuitively makes sense.

Visualizing user effects



In the aforementioned histogram, we can clearly see that user effect is rather strong, and rating variability is high from user to user. While some users are very generous with their ratings, others are rather conservative. For predicting ratings, user effect therefore should be taken into account.

Visualizing the time effect on ratings



While it does appear that older movies enjoyed slightly better ratings, the effect of age is not strong.

Creating a model

We want to create a model that incorporates user effect, movie effect and genre effect. The time effect has been dropped because it does not look significant.

Creating Loss Function

We start by creating an RMSE function that we will use to measure the accuracy of our models.

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

Building the Model

First, we create a very simple model using just the average and the movie effect.

```
# Calculating overall average

mu <- mean(edx$rating)

# Calculating Naive RMSE using "Just the average"

just_the_avg <- RMSE(validation$rating,mu)
print(just_the_avg)
```

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

Then we factor in the movie bias

```
# Calculating movie bias

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

# Predicted rating using average and movie bias

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


movie_bias_model <- RMSE(validation$rating,predicted_ratings)
print(movie_bias_model)
```

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

After factoring movie bias, we see that our prediction accuracy has improved. Let's see if it improves even further if we take user bias into consideration.

```
# Calculating user bias

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

predicted_ratings_1 <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

movie_and_user_bias_model <- RMSE(validation$rating,predicted_ratings_1)
print(movie_and_user_bias_model)
```

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

As we have observed from our previous analysisi that ratings can vary based on genre, so we will consider the genre bias as well.

```
# Calculating genre bias

genre_avgs <- edx %>% left_join(movie_avgs,by='movieId') %>% left_join(user_avgs,by='userId') %>% group_

predicted_ratings_2 <- validation %>%
  left_join(movie_avgs,by='movieId') %>% left_join(user_avgs,by='userId') %>% left_join(genre_avgs,by='

movie_user_genre_bias_model <- RMSE(validation$rating,predicted_ratings_2)
print(movie_user_genre_bias_model)
```

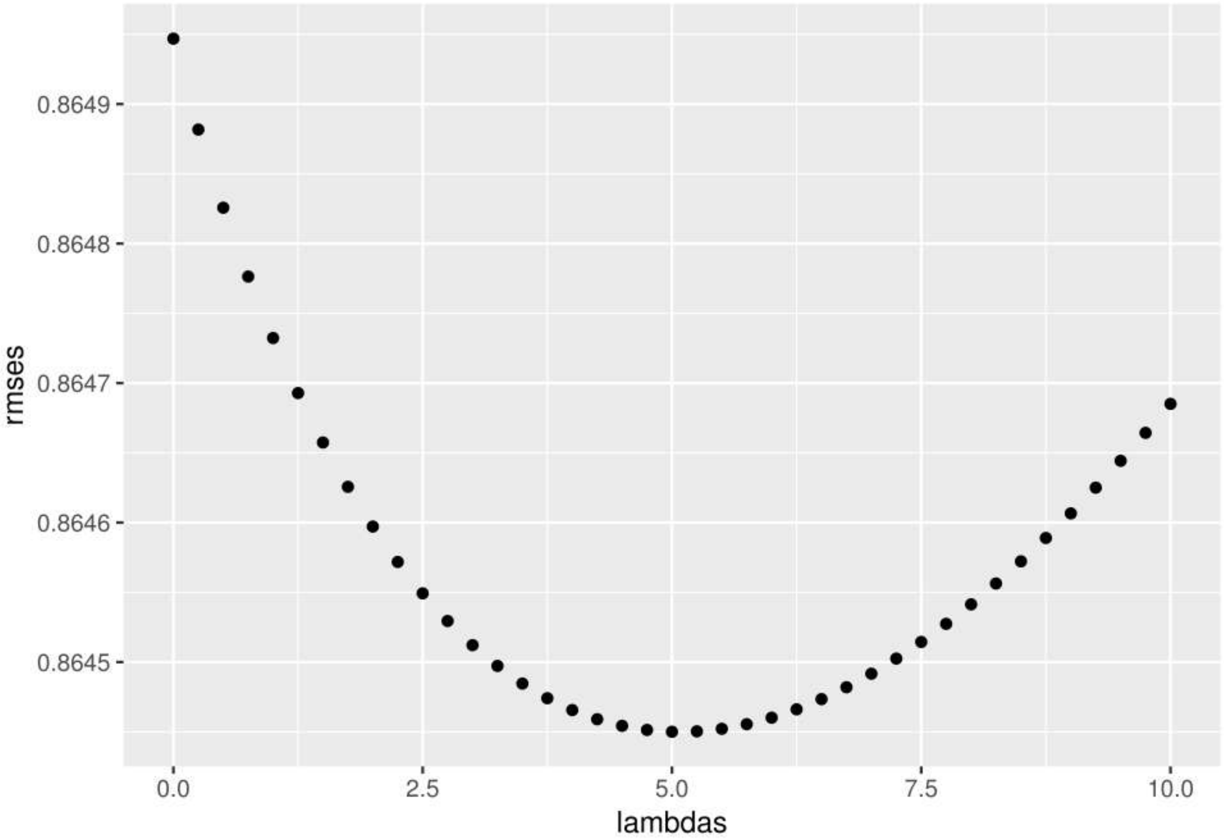
```
## [1] 0.8649469
```

However, we might be able to improve the accuracy of our model even further if we use the regularization technique.

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

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

    mu_reg <- mean(edx$rating)

    b_i_reg <- edx %>%
        group_by(movieId) %>%
        summarize(b_i_reg = sum(rating - mu_reg)/(n()+l))

    b_u_reg <- edx %>%
        left_join(b_i_reg, by="movieId") %>%
        group_by(userId) %>%
        summarize(b_u_reg = sum(rating - b_i_reg - mu_reg)/(n()+l))

    b_g_reg <- edx %>% left_join(b_i_reg,by='movieId') %>% left_join(b_u_reg,by='userId') %>% group_by(
    predicted_ratings_b_i_u_g <-
        validation %>%
        left_join(b_i_reg, by = "movieId") %>%
        left_join(b_u_reg, by = "userId") %>%
        left_join(b_g_reg,by='genres') %>%
        mutate(pred = mu_reg + b_i_reg + b_u_reg+b_g_reg) %>%
        .$pred

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

Let's look at the optimum Lamda

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



```
lambdas[which.min(rmses)]
```

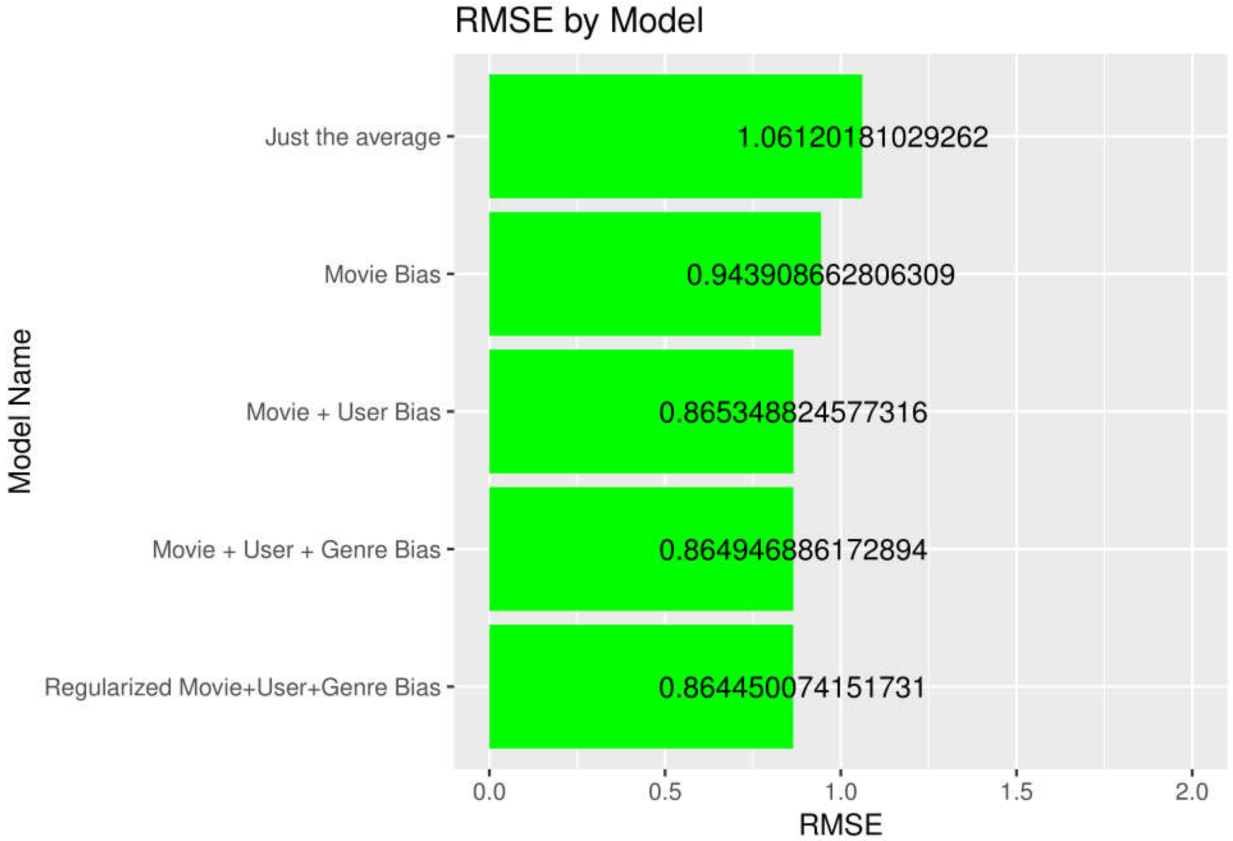
```
## [1] 5
```

```
# RMSE for regularized model that incorporates genre, user and movie bias

regularized_model_with_movie_genre_user_bias <- min(rmses)
print(regularized_model_with_movie_genre_user_bias)
```

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

Results

There seems to be dramatic improvement over the "just the average approach" when we use movie and user bias into account. And slight improvements when we incorporate genre bias and regularization techniques.



Conclusion

By using regularization and modelling user bias, movie bias and genre bias, it is possible to reduce the RMSE to 0.8644501, which meets the "RMSE <= 0.8649" requirement for full 25 points. The time effect was not modelled because from the initial analysis it became apparent that the time bias is not as strong as the other three. However the RMSE can be reduced even further by using advanced techniques like ensemble learning and model stacking. But for a large dataset, those techniques cannot be used from a desktop/laptop computer due to insufficient memory. The next step would be to use a cloud computing platform that supports machine learning, like Azure or Google AI Platform and use ensemble learning and model stacking to achieve more accurate results.