MovieLens Project

Tobiloba Oyediran

5/24/2021

OVERVIEW

The MovieLens 10M dataset is a compilation of ratings given to movies by users who have watched the movies.

The objective of this project is to develop an algorithm which can predict the rating a user is likely to give a movie. To achieve this a prediction model is developed and progressively improved using a large part (90%) of the MovieLens dataset to train the model. This is split into a training set (for training the model) and test set (for testing the model after each improvement), using the root-mean-square-error, RMSE, as the measure of performance of the model.

The incremental steps taken in developing and improving the model are: 1 - Guessing the rating 2 - Assuming average rating 3 - Incorporating influence of the movie (movie effect) on the rating 4 - Incorporating biases of individual users (user bias) on the rating 5 - Regularizing movie effect and user bias

A significant improvement in the RMSE (from 1.94146770 at step 1, to 0.86473761 at step 5) obtained.The final improved model is then used to predict the ratings in the remaining part (10%) of the dataset. The final RMSE obtained for this prediction is 0.86481774

ANALYSIS

After Downloading and Tidying the data

Inspecting the dataset

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     231      5 838983392          Dumb & Dumber (1994)
## 4:      1     292      5 838983421               Outbreak (1995)
## 5:      1     316      5 838983392               Stargate (1994)
## 6:      1     329      5 838983392 Star Trek: Generations (1994)
##                            genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:                         Comedy
## 4:  Action|Drama|Sci-Fi|Thriller
## 5:       Action|Adventure|Sci-Fi
## 6: Action|Adventure|Drama|Sci-Fi
```

All observations are now in rows and the variables in the dataset are in columns. The dataset is tidy and ready for analysis.

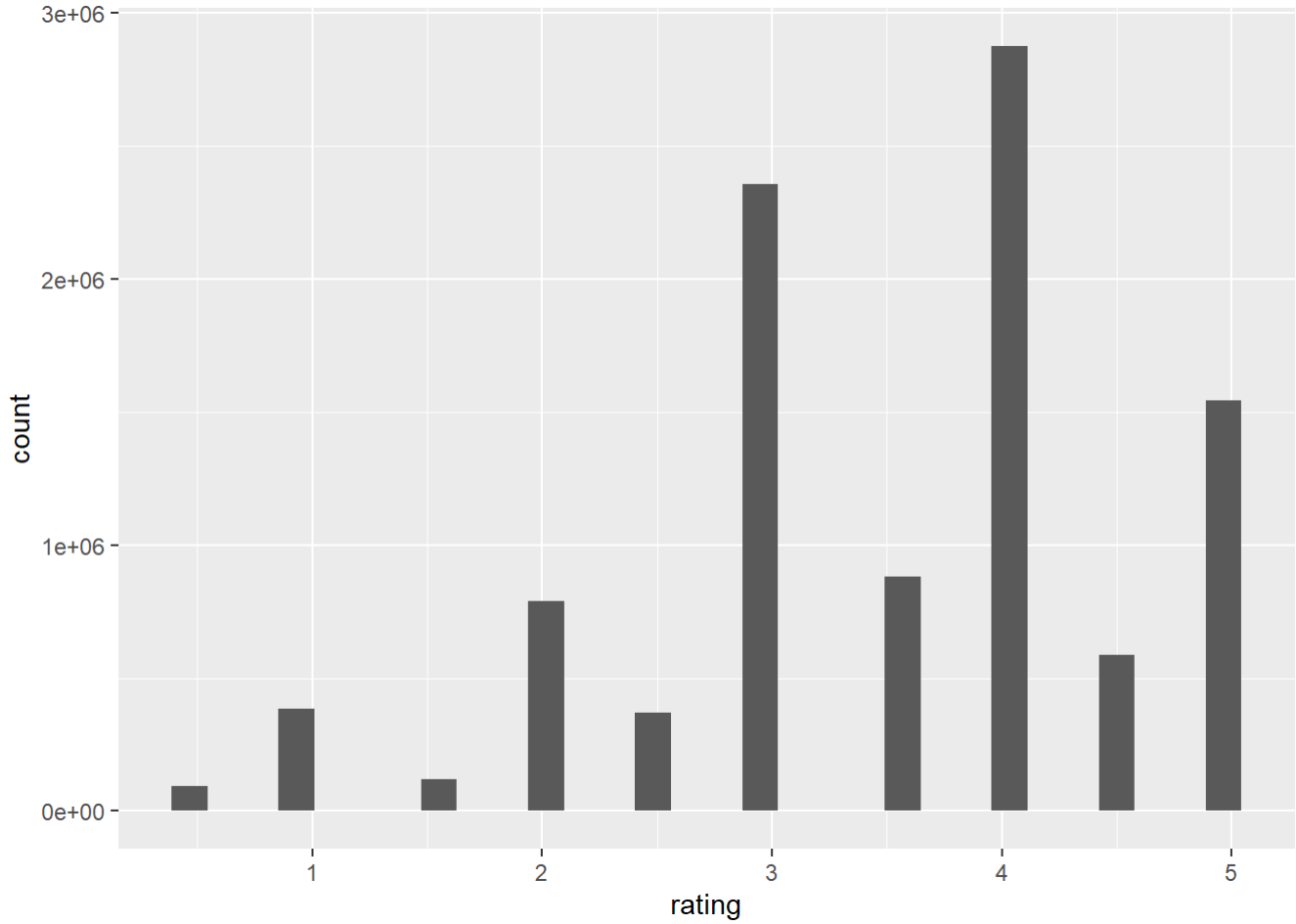
Data Exploration

```
## Classes 'data.table' and 'data.frame':   10000054 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 231 292 316 329 355 356 362 364 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983392 838983421 838983392 838983392 838984474 8
38983653 838984885 838983707 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb & Dumber (1994)" "Outbreak
(1995)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Comedy" "Action|Drama|Sci-Fi|
Thriller" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

The movielens dataset contains the **10000054 observations**. There are 6 variables: **userId** (ID assigned to user rating the movie), **movieId** (ID assigned to the movie being rated), **rating** (the rating given by user), **timestamp** (numeric value of time when rating was given), **title** (movie title, with year of release in parenthesis ()) and **genres** (all the genres the movie belongs to, each genre separated by a pipe |)

```
## [1] 0.5 5.0
```

The rating given to a movie ranges between 0.5 and 5.0



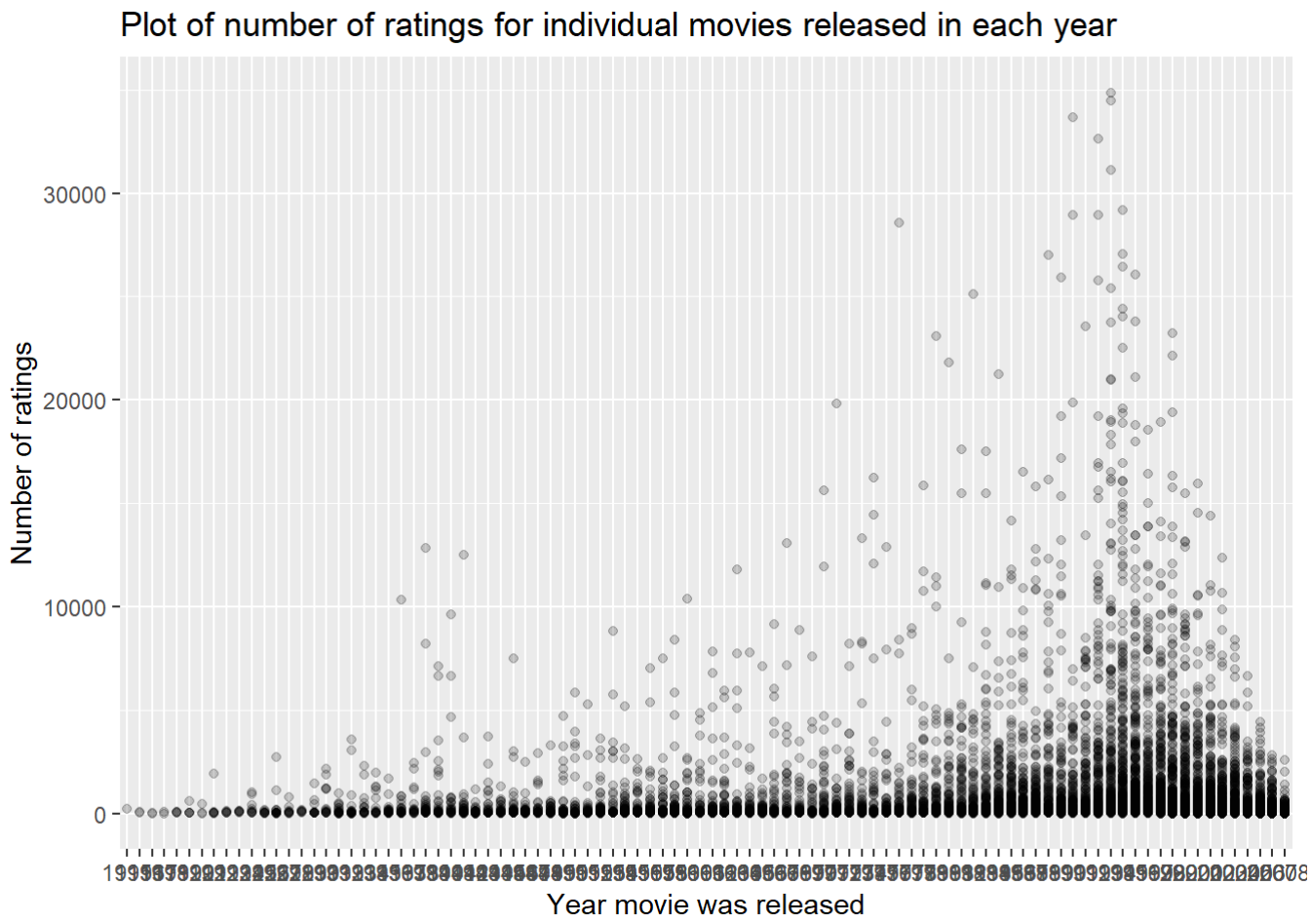
It appears as though there is a general bias towards giving a full-star (integer) rating rather than half-star (fractional) rating. 4.0, followed by 3.0 and 5.0 are three most common ratings.

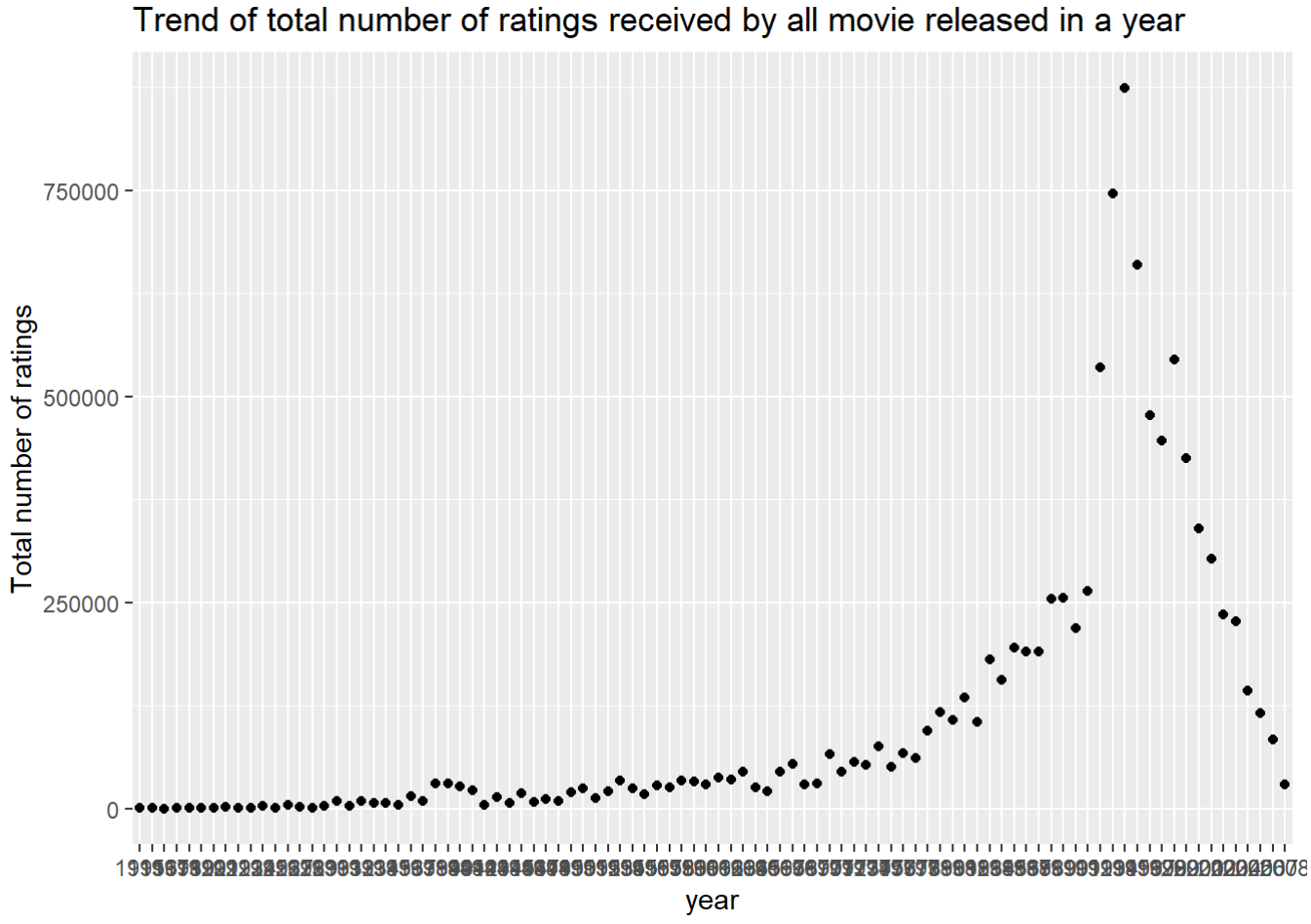
Inspecting the range of years the movies were released

```
## [1] "1915" "2008"
```

The dataset contains movies released from 1915 to 2008.

Visualizing the general trend of ratings over the years





From the visualizations above, the number of ratings for movies generally increased over time, and then declined sharply for movies recently released.

```
## # A tibble: 10 x 2
##    year  n_ratings
##    <chr>     <int>
##  1 1995     874436
##  2 1994     746042
##  3 1996     659425
##  4 1999     543990
##  5 1993     534899
##  6 1997     477463
##  7 1998     446739
##  8 2000     425218
##  9 2001     339508
## 10 2002     302452
```

The year with the highest total number of ratings is 1995, with 874436 ratings. The top 10 years with the highest total number of ratings fall within the early 1990s to early 2000s. It is likely that the development of machine learning field and the advent of recommender systems contributed to the rise in the number of ratings over the years. The latest movies may not have been watched by enough users for them to have as many ratings as is expected.

Developing a prediction model

The edx data is split to create a test set (20%), and a training set (80%)

Function to calculate root-mean-square error (RMSE

```
RMSE <- function(predicted_value, true_value) {
  sqrt(mean((predicted_value - true_value)^2))
}
```

Building the Recommender System Algorithm

Step 1: Randomly guessing the rating

```
## [1] 1.941945
```

The RMSE is almost 2

Step 2: Predicting the average rating for all movies

```
## [1] 1.060704
```

We see a significant improvement in RMSE when predicting average compared to when just guessing, (RMSE dropped from 1.941945 to 1.060704) as the mean is more representative of the data than just a random guess.

Step 3: Trying to improve prediction by incorporating (possible) movie effect

To make up for the error in predicting the average rating by factoring in the effect of individual characteristics: we use the average error in prediction for each movie as the movie effect

\(movieEffect = sum(trueRating - averageRating)/N\) where N = number of ratings for the movie

```
## [1] 0.9437144
```

We see more improvement in RMSE: from 1.060704 to 0.9437144, when movie effect is accounted for. This shows that there is indeed a “movie effect”

Step 4: Improve prediction by further incorporating (possible) user bias

To make up for the error in predicting the average rating by factoring in the individual user’s personal preferences concerning movies: we use the average error in prediction (after including the movie effect calculated above) for each user as the user bias

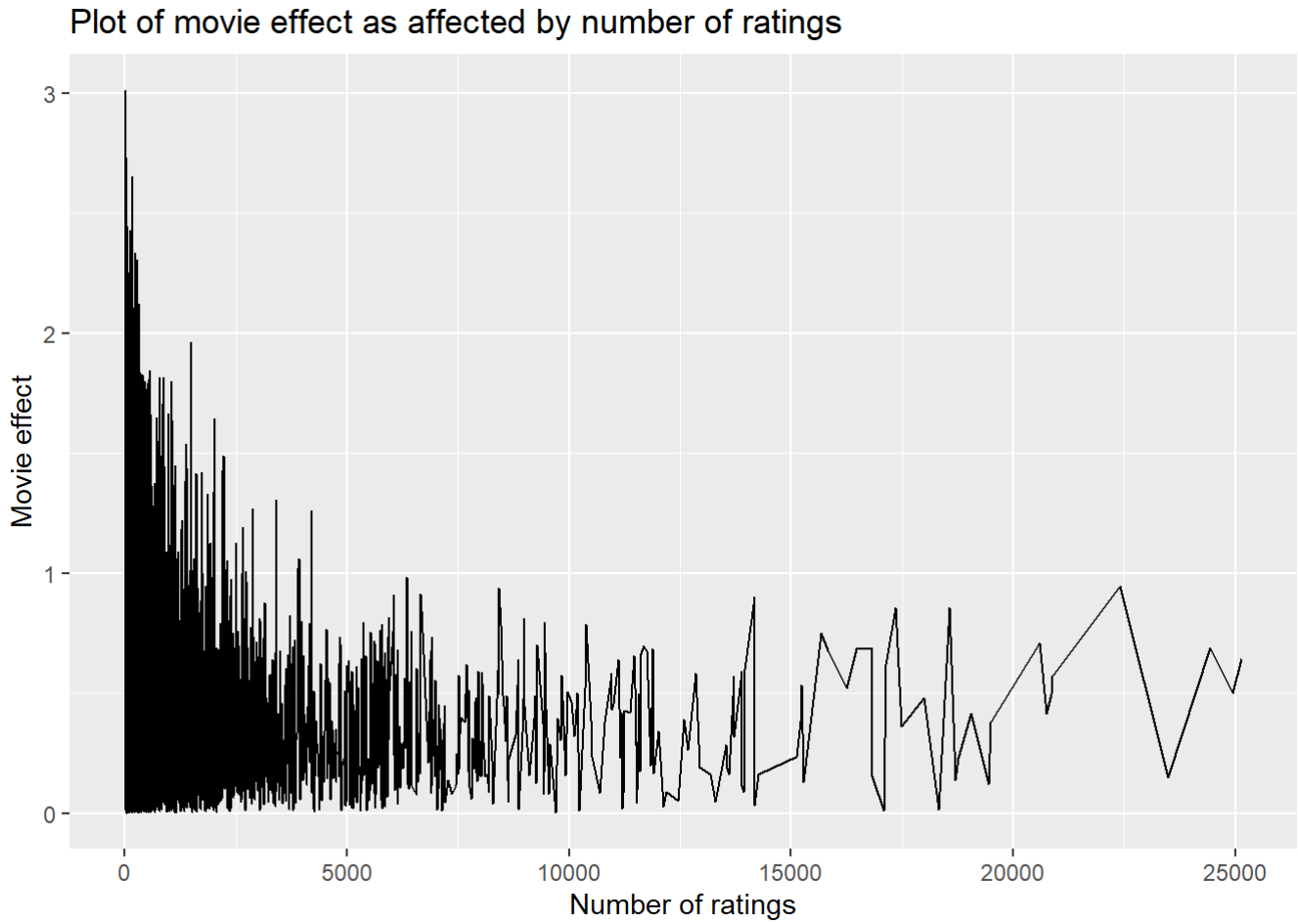
\(userBias = sum(trueRating - (averageRating + movieEffect))/N\) where N = number of ratings given by user

```
## [1] 0.8661625
```

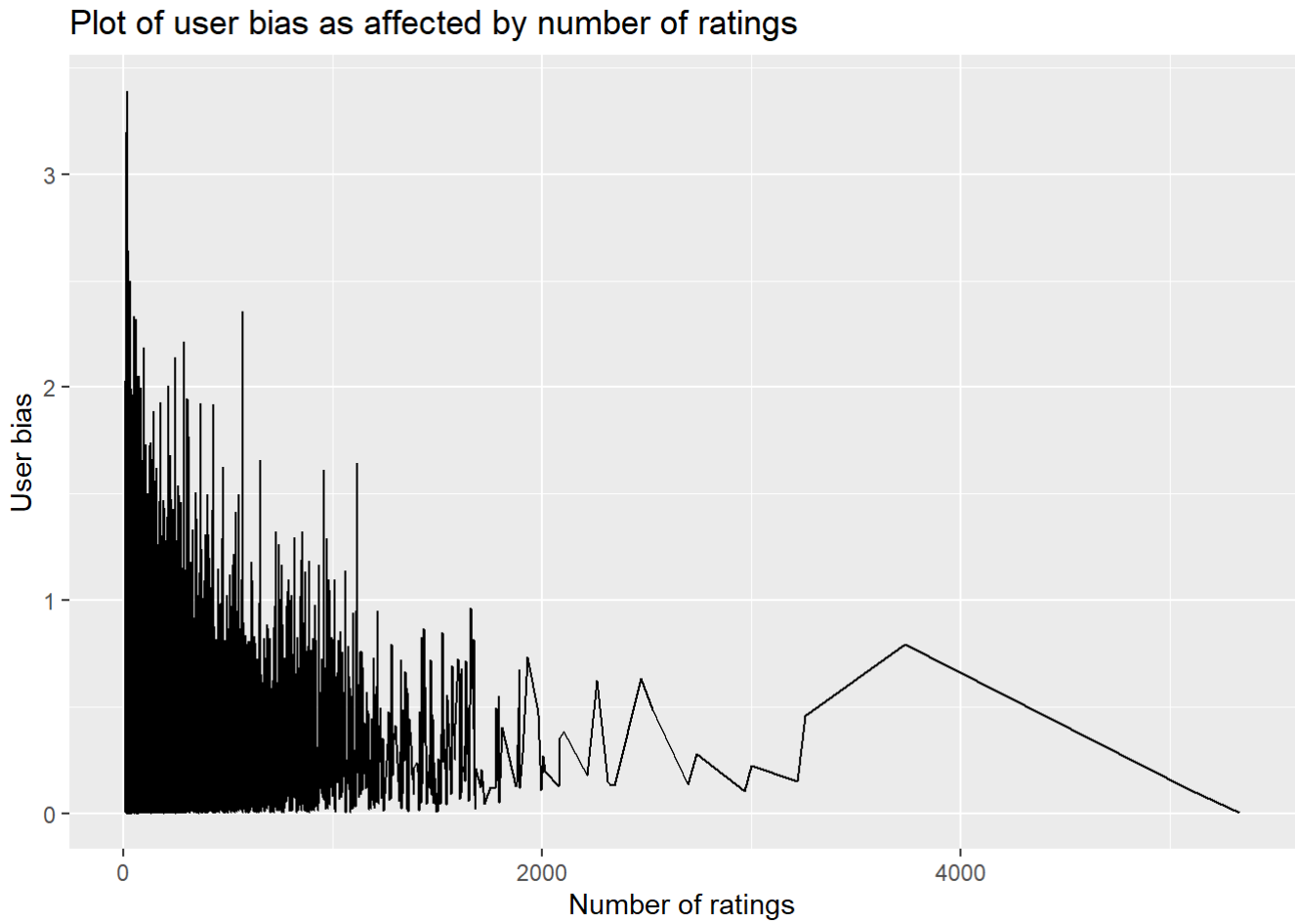
We see more improvement in RMSE: from 0.9437144 to 0.8661625, when both movie effect and user bias are accounted for. This shows that individual user’s bias also affects the rating given to a movie.

Exploring the data further

To see if the number of ratings affects the difference between the true rating and the average rating.



Generally, the movie effect tends to get closer to zero, i.e. the error in prediction due to individual movie characteristics tends to diminish to almost zero as the number of ratings increases.



Also, the user bias generally tends to get closer to zero, i.e. the error in prediction due to individual user's movie preferences tends to diminish to almost zero as the number of ratings increases.

Applying regularization to the movie effect and user bias by adding an arbitrary number, lambda (l), to the number of ratings for each movie while calculating movie effect, and for each user while calculating the user bias will reduce the error in prediction for movies that are unpopular (those that have few ratings)

Step 5: Regularizing the effect of unpopular movies

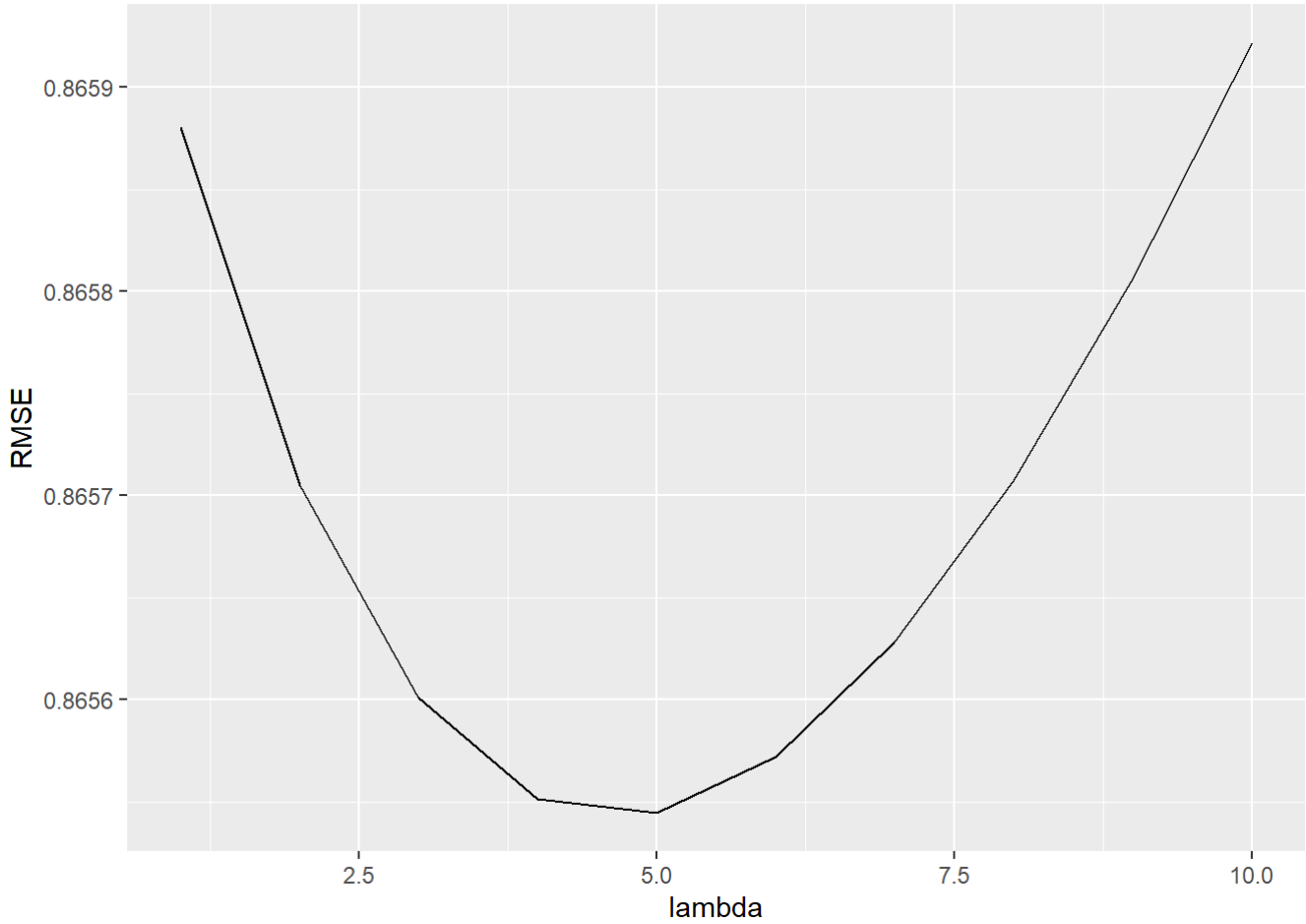
Select a range of lambdas to find which value minimizes the RMSE

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Create function to calculate regularized RMSE for a given value of lambda using

\(movieEffectReg = sum(trueRating - averageRating)/(N + lambda)\) \(userBiasReg = sum(trueRating - (averageRating + movieEffect))/(N + lambda)\)

Plot of RMSEs of the regularized model for each lambda value in the range selected



From the plot, it appears that 5 is the value of lambda that minimizes the RMSE when regularization is applied to movie effect and user bias in the prediction model

The value of lambda that best minimizes the RMSE

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

The value of lambda that minimizes the RMSE is 5

The best RMSE (when lambda of 5 is used)

```
## [1] 0.8655444
```

This also gives an improvement in RMSE

Step 6: Apply the improved model in Step 5 to predict the values in the validation data set

Using the best value of lambda (5); with edx as the training data for the model:

1. Recalculate the regularized movie effect and user bias, and Predict rating in the validation data

2. Calculate the final RMSE

```
## [1] 0.8648177
```

The RMSE obtained using the edx dataset as the training data, and the validation data for testing the final model is 0.8648177. This shows an improvement over the 0.8655444 obtained when we used the data partitions (train_set for training, and test_set for testing) from the edx dataset.

RESULTS

RMSEs obtained for each step of model training, and calculating the % improvement in model performance

```
# Update the results table
results <- bind_rows(results, data.frame(method = "Improved model applied to the validation s
et",
                                         RMSE = RMSE_validation))
# Calculate percentage improvement in model performance
results <- mutate(results, "% improvement" = (RMSE[1] - RMSE)*100 / RMSE[1])
results <- bind_cols(data.frame(step = c(1:6)), results)
results
```

```
##   step                                        method      RMSE % improvement
## 1    1                                   Just a guess 1.9419448       0.00000
## 2    2                              Predicting average 1.0607045      45.37927
## 3    3              Model incoporating movie effect 0.9437144      51.40364
## 4    4        Model with movie effect and user bias 0.8661625      55.39716
## 5    5           Improved model with regularization 0.8655444      55.42899
## 6    6 Improved model applied to the validation set 0.8648177      55.46641
```

From the results table, it is evident that:

1. Predicting the average overall rating performed much better that the random guessing. This shows that the overall average is more representative of the dataset than a random guess.
2. There is incremental improvement in performance of the model as other influencing factors such as movie effect and user bias are introduced into the model.
3. Comparing the performance of the model in Steps 5 and 6, the model performed better when a larger training set is used (edx is larger than train_set). This suggests that the larger the training dataset, the lower the error of prediction using this model.

CONCLUSION

The model performance has been significantly improved by incorporating some factors such as movie effect and user bias that have been seen to affect how users rate movies. Regularizing these effects also provided additional improvement.

Limitation of this work

This report does not explore the effect of other factors such as the genre of the movie, or the time gap between when the movie was released and when it was rated. These factors could also significantly affect how users rate movies and could be useful for improving the prediction model.

Future work

To further improve the performance of the prediction model, an exploration of the effects of movie genre and time gap between movie release and rating could be a way to go. Also, applying more sophisticated algorithms such as nearest neighbor or random forest, as well as creating an ensemble of multiple algorithms, may produce far greater improvements for the prediction model.