Movie Lens Final Project

Matt Harvill

7/13/2021

INTRODUCTION

In October of 2006 Netflix offered a prize of one million dollars to anyone or group that could improve their movie recommendation algorithm by 10%. While Netflix does not offer public access to their data, the GroupLens research lab generated their own database with over 20 million ratings for over 27,000 movies by more than 138,000 users. This capstone project will create a movie recommendation algorithm inspired by the Netflix challenge using the GroupLens data.

OVERVIEW

The recommendation algorithm will attempt to predict ratings users will give movies they have not rated or seen. It will account for the user's own personal biases, the general popularity of certain movies and genres, while accounting for weighted predictions of obscure movies that are subsequently rated very high or very low as well as popular movies that are rated more often.

Data Loading

Create edx set, validation set (final hold-out test set)

```
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")

library(tidyverse)
library(caret)
library(data.table)
```

Note: this process could take a couple of minutes

MovieLens 10M dataset:

https://grouplens.org/datasets/movielens/10m/

```
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")
```

http://files.grouplens.org/datasets/movielens/ml-10m.zip

```
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))



movielens <- left_join(ratings, movies, by = "movieId")
```

if using R 4.0 or later:

```
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
```

Validation set will be 10% of MovieLens data

```
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
```

Make sure userId and movieId in validation set are also in edx set

```
removed <- anti_join(temp, validation)
```

Add rows removed from validation set back into edx set

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

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

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

METHODS & ANALYSIS

Multiple biases will be identified and accounted for in the final model and large outliers to the data will be accounted for and regularized. The Residual Mean Squared Error, or RMSE, will be the measure of the accuracy of the prediction algorithm. An overview of the data will help identify these outliers and biases in order to construct the prediction models.

Data Analysis

first 7 rows with headers

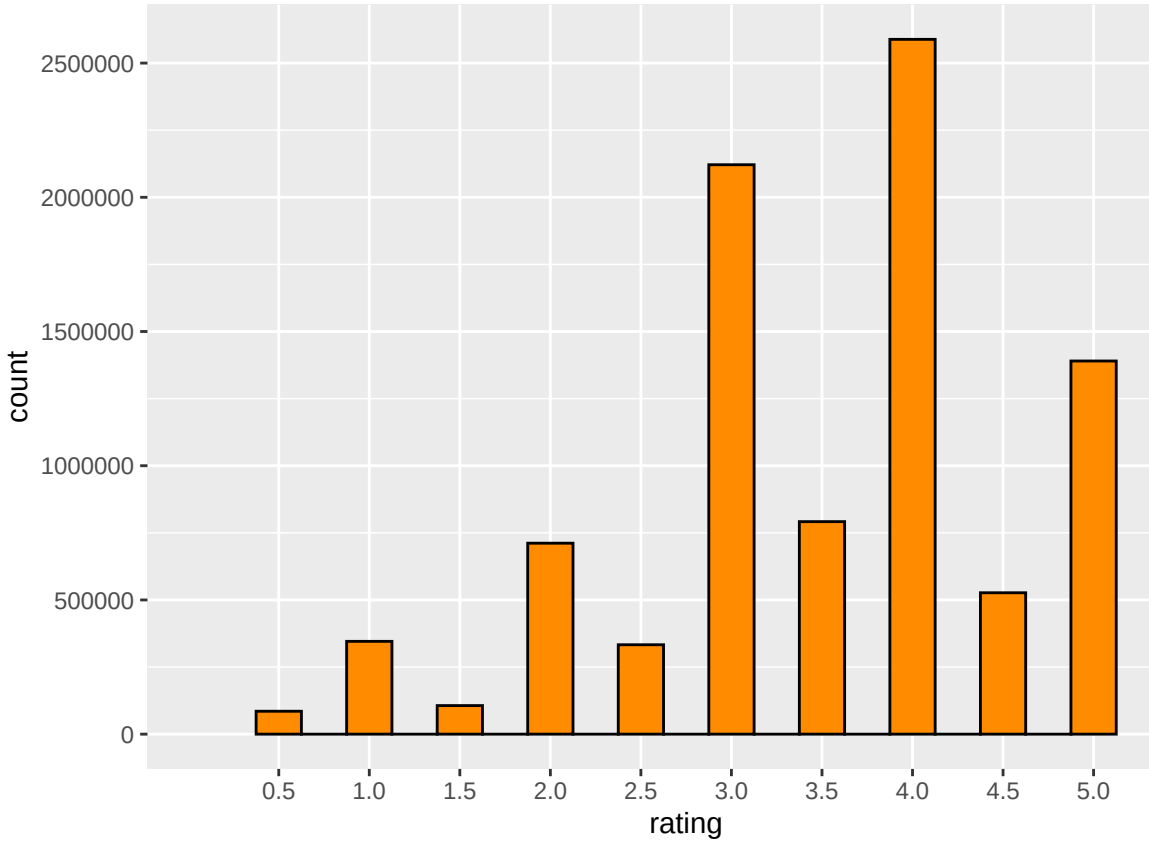
```
##    userId movieId rating timestamp                          title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                            genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

basic summary statistics for each column

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

number of unique users and movies in the dataset

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



ratings distribution

five most given ratings in order from most to least

```
## Selecting by count
```

```
## # A tibble: 5 x 2
##   rating   count
##    <dbl>   <int>
## 1    4   2588430
## 2    3   2121240
## 3    5   1390114
## 4    3.5  791624
## 5    2    711422
```

We can see that movies are generally rated higher than lower and on the whole number rather than the half.

top 10 movies ranked by the number of ratings

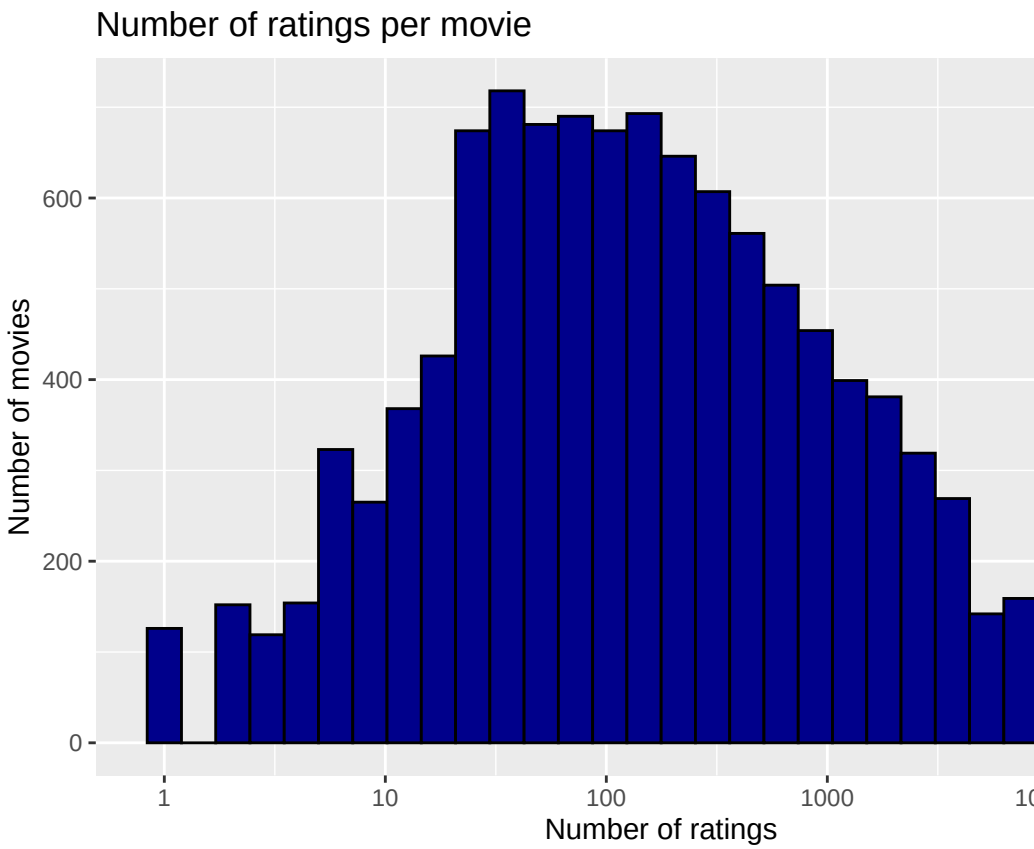
```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                     count
##      <dbl> <chr>                                     <int>
##  1     296 Pulp Fiction (1994)                       31362
##  2     356 Forrest Gump (1994)                       31079
##  3     593 Silence of the Lambs, The (1991)          30382
```

```
##  4     480 Jurassic Park (1993)                                       29360
##  5     318 Shawshank Redemption, The (1994)                           28015
##  6     110 Braveheart (1995)                                          26212
##  7     457 Fugitive, The (1993)                                       25998
##  8     589 Terminator 2: Judgment Day (1991)                          25984
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
## 10     150 Apollo 13 (1995)                                           24284
## # ... with 10,667 more rows
```

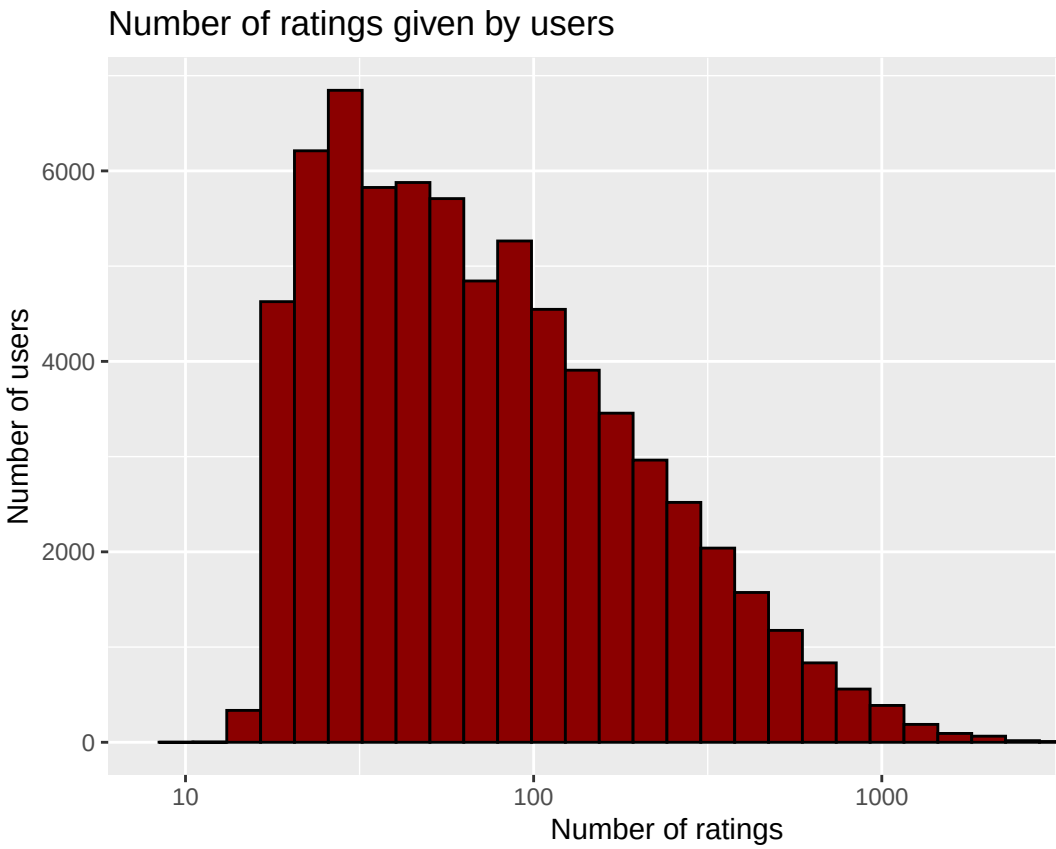
The most rated movies tend to be the "blockbuster" films

average rating across all movies

```
##   mean(rating)
## 1     3.512465
```



distribution of movie ratings



distribution of user ratings

Some movies are rated much more often than others while some users are much more active in the rating of films. Both of these effects will need to be modeled and factored in to the final prediction algorithm.

Building the Recommendation System

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

Residual Mean Square Error formula for testing accuracy of predictions

Average movie rating model

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

average of all ratings across all users

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

```
naive_rmse <- RMSE(validation$rating, mu)
naive_rmse
```

predict all unknown ratings with mu

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

This number is the focus. Reducing the RMSE, the measure of the accuracy of the prediction, is the goal of the algorithm.

```
rmse_results <- tibble(method = "Just the average", RMSE = naive_rmse)
```

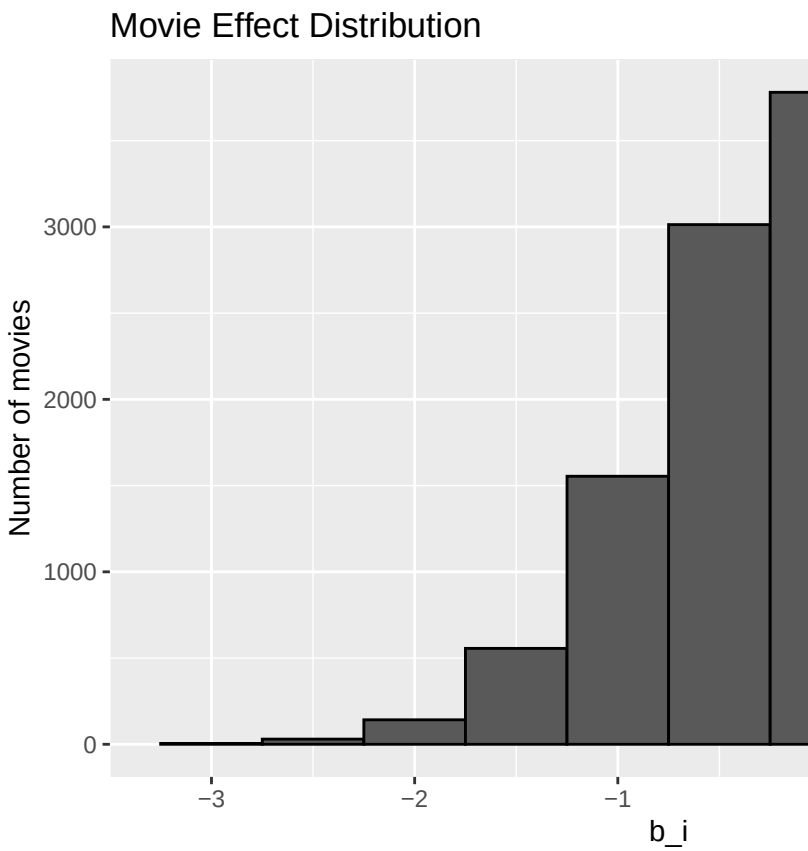
create a table to store results of prediction approaches

table showing naive RMSE prediction average

method	RMSE
Just the average	1.061202

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

model accounting for movie effect (b_i)



plot the number of movies with computed b_i

More popular movies are rated more frequently by users. This movie effect will need to be accounted for.

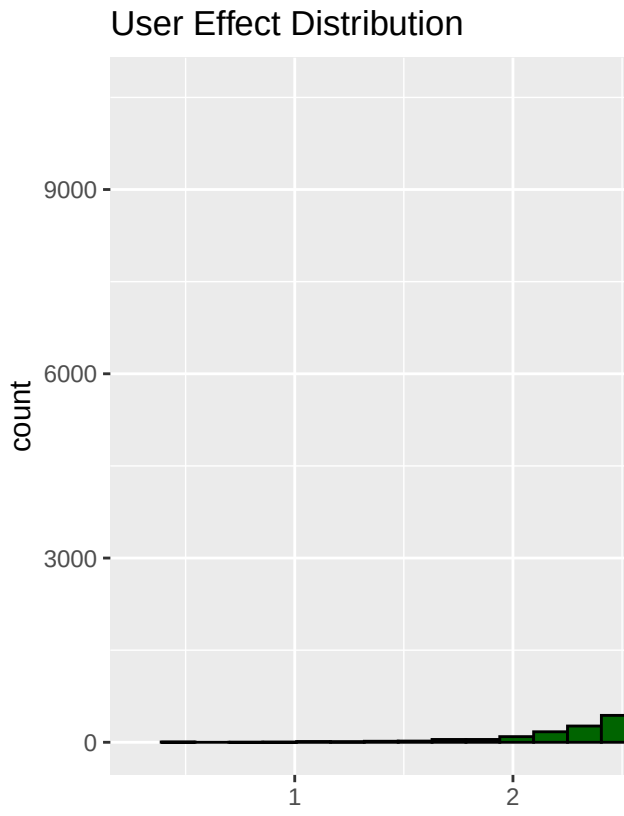
```
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i
model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie Effect Model",
                                 RMSE = model_1_rmse))
```

test and save RMSE results

table showing movie effect model results

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087

Accounting for the movie effect has reduced the RMSE.



plot showing user effect for users with more than 100 ratings

Some users are more active in the rating of movies on the platform. This user effect will need to be penalized in order to normalize effect it has on the prediction.

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

model accounting for user effect (b_u) + movie effect

```
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred
model_2_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method = "Movie + User Effects Model",
                                     RMSE = model_2_rmse))
```

test and save new RMSE results

table showing movie + user effect model results

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488

The RMSE has been reduced further accounting for these two effects.

Regularization of movie + user effect model

Obscure films, sometimes "art" films or little seen genre films can be disproportionately weighted in the prediction by relatively high ratings from enthusiastic fans of such movies. These outliers will be penalized and the prediction model normalized.

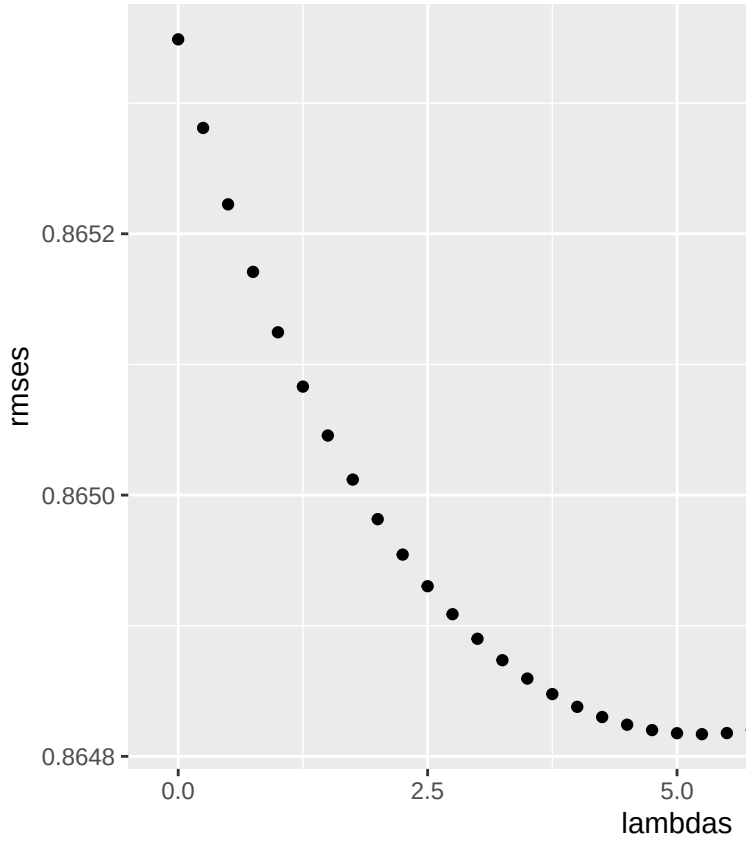
```
lambdas <- seq(0, 10, 0.25)
```

lambda is a tuning parameter, chosen by cross-validation

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

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

below code may take several minutes to run



plot lambdas and RMSEs to select optimal lambda

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

find optimal lambda

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

```
rmse_results <- bind_rows(rmse_results,
                          tibble(method = "Regularized Movie + User Effect Model",
                                 RMSE = min(rmses)))
```

regularized model accounting for movie + user effect

table showing regularized movie + user effect model results

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087

method	RMSE
Movie + User Effects Model	0.8653488
Regularized Movie + User Effect Model	0.8648170

A further reduction of the RMSE when regularizing the model

Matrix factorization of genres to refine prediction

The movies in the dataset are categorized by genre, and many have more than one genre attached to them. By separating the multi-genre films and using individual genres as a factor for prediction the RMSE can be further refined.

split movies with multiple genres in the train set and validation set

```
genre_split_edx <- edx %>% separate_rows(genres, sep = "\\|")
genre_split_validation <- validation %>% separate_rows(genres, sep = "\\|")
```

below code may take several minutes to run

view genre split

```
## # A tibble: 6 x 6
##   userId movieId rating timestamp title            genres
##    <int>   <dbl>  <dbl>     <int> <chr>            <chr>
## 1      1     122      5 838985046 Boomerang (1992) Comedy
## 2      1     122      5 838985046 Boomerang (1992) Romance
## 3      1     185      5 838983525 Net, The (1995)  Action
## 4      1     185      5 838983525 Net, The (1995)  Crime
## 5      1     185      5 838983525 Net, The (1995)  Thriller
## 6      1     292      5 838983421 Outbreak (1995)  Action
```

Checking that genre split has been acheived

add genre effect to prediction model

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

  b_i <- genre_split_edx %>%
    group_by(movieId) %>%
    summarise(b_i = sum(rating - mu)/(n()+l))
```

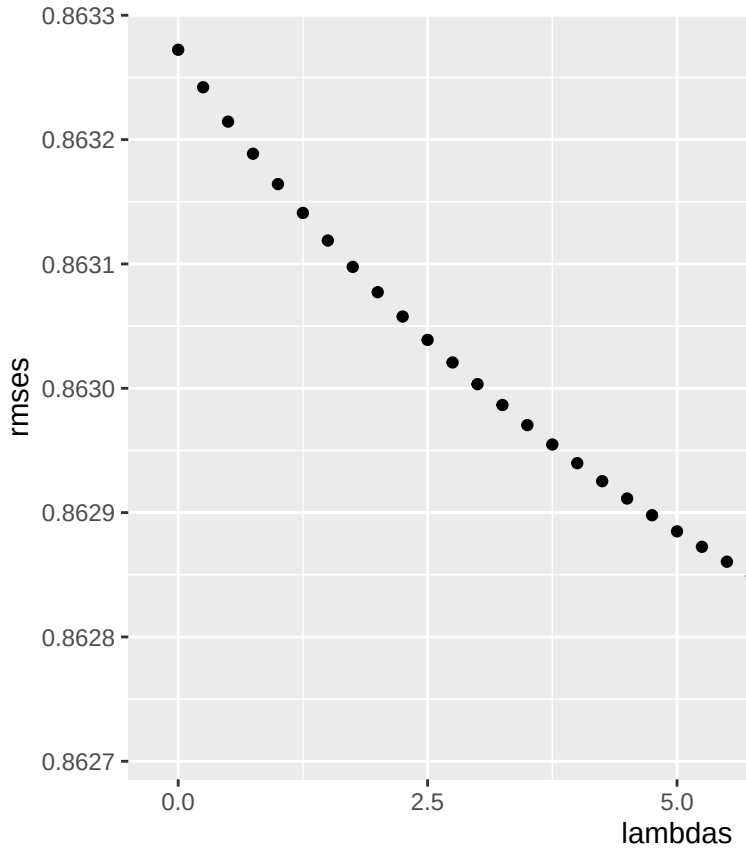
```
  b_u <- genre_split_edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarise(b_u = sum(rating - b_i - mu)/(n()+l))

  b_g <- genre_split_edx %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    group_by(genres) %>%
    summarise(b_g = sum(rating - b_i - b_u - mu)/(n()+1))

  predicted_ratings <- genre_split_validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    mutate(pred = mu + b_i + b_u + b_g) %>%
    .$pred

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

below code may take several minutes to run



plot lambdas and RMSEs to select optimal lambda

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

find optimal lambda

```
## [1] 10
```

```
rmse_results <- bind_rows(rmse_results,
                          tibble(method = "Regularized Movie + User + Genre Effect Model",
                                 RMSE = min(rmses)))
```

regularized model accounting for movie + user + genre effect At this point the biases of genre, user and movie effect have been factored and regularized

RESULTS

final RMSE results

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Regularized Movie + User Effect Model	0.8648170
Regularized Movie + User + Genre Effect Model	0.8627135

The table shows a marked improvement from the naive RMSE just accounting for the average rating of all the films by all users

CONCLUSION

Using machine learning techniques to train the movie prediction algorithm a RMSE of 0.8627135 was achieved. This could possibly be improved further by testing to see if the year the movie was made has a biased effect, or using more complex models and averaging the results. However given the computational requirements for these models on a dataset this size, this has been left for future work.