Movie Lens Final Project

Matt Harvill

7/13/2021

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

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

Note: this process could take a couple of minutes

```
## Loading required package: tidyverse

## Warning: package 'tidyverse' was built under R version 4.0.5

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.6
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'readr' was built under R version 4.0.5

## Warning: package 'purrr' was built under R version 4.0.5

## Warning: package 'dplyr' was built under R version 4.0.5

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

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

```
## Loading required package: caret

## Warning: package 'caret' was built under R version 4.0.5
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

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

```
## Loading required package: data.table

## Warning: package 'data.table' was built under R version 4.0.5

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

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

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")
```

if using R 4.0 or later:

```r
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")
```

Validation set will be 10% of MovieLens data

```r
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```r
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

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

Add rows removed from validation set back into edx set

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

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

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

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

Data Analysis

first 7 rows with header

```
##    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

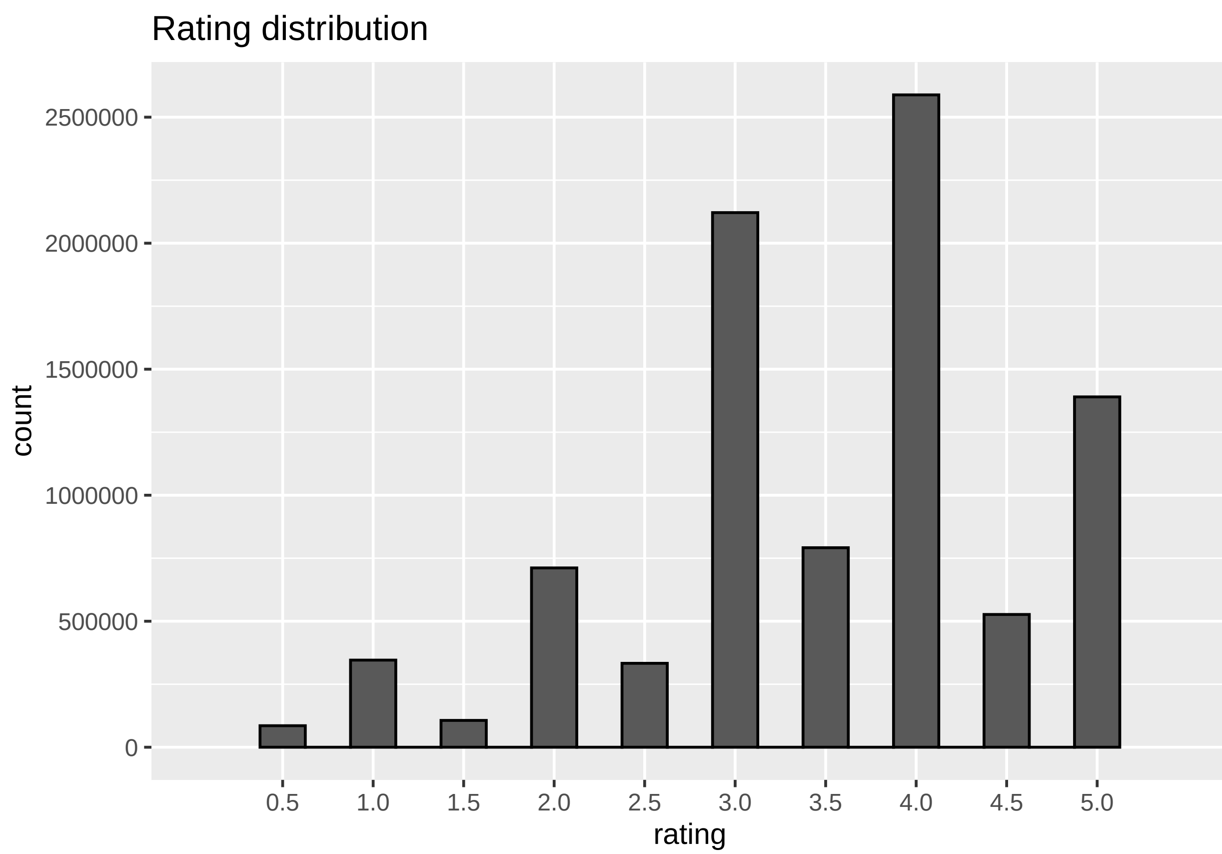
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 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

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

ratings distribution

```
## Warning: Continuous limits supplied to discrete scale.
## Did you mean ‘limits = factor(...)‘ or ‘scale_*_continuous()‘?
```



five most given ratings in order from most to least

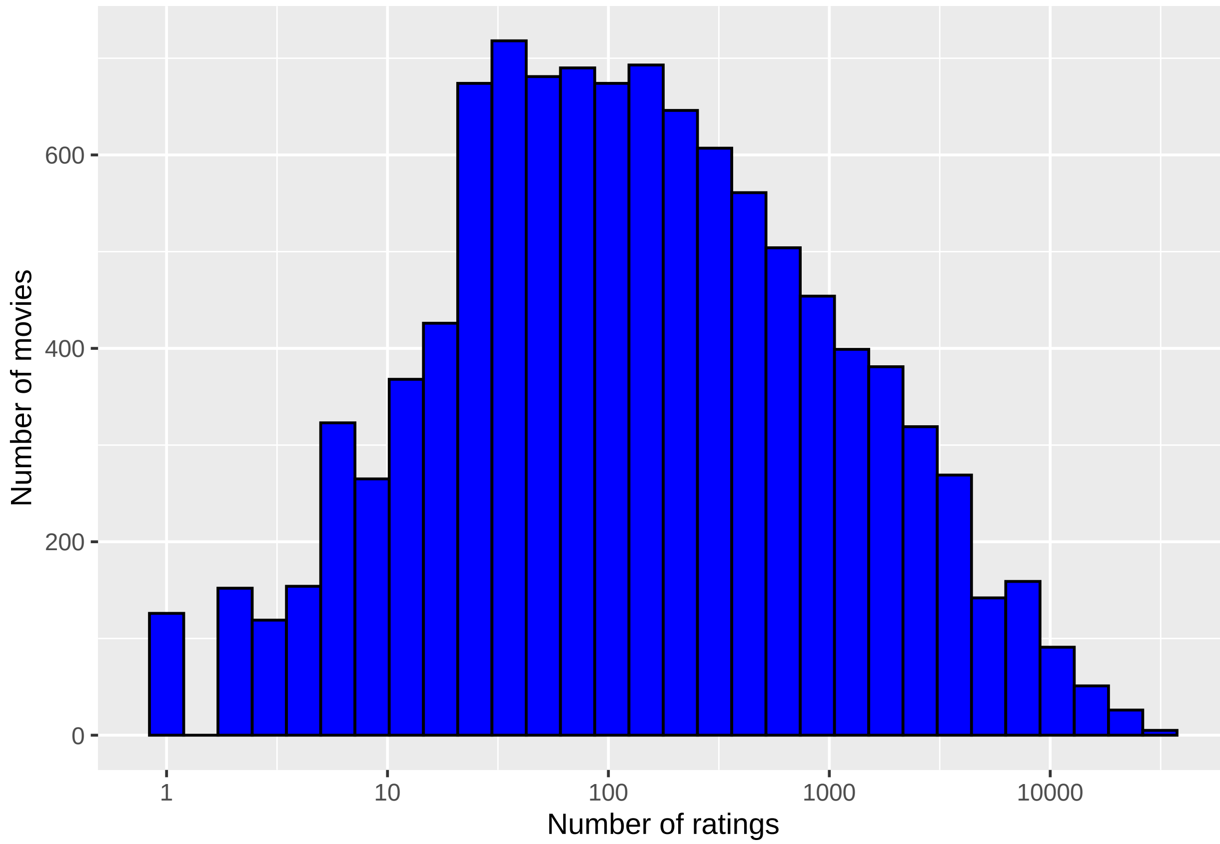
```
## Selecting by count
```

```
## # 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
```

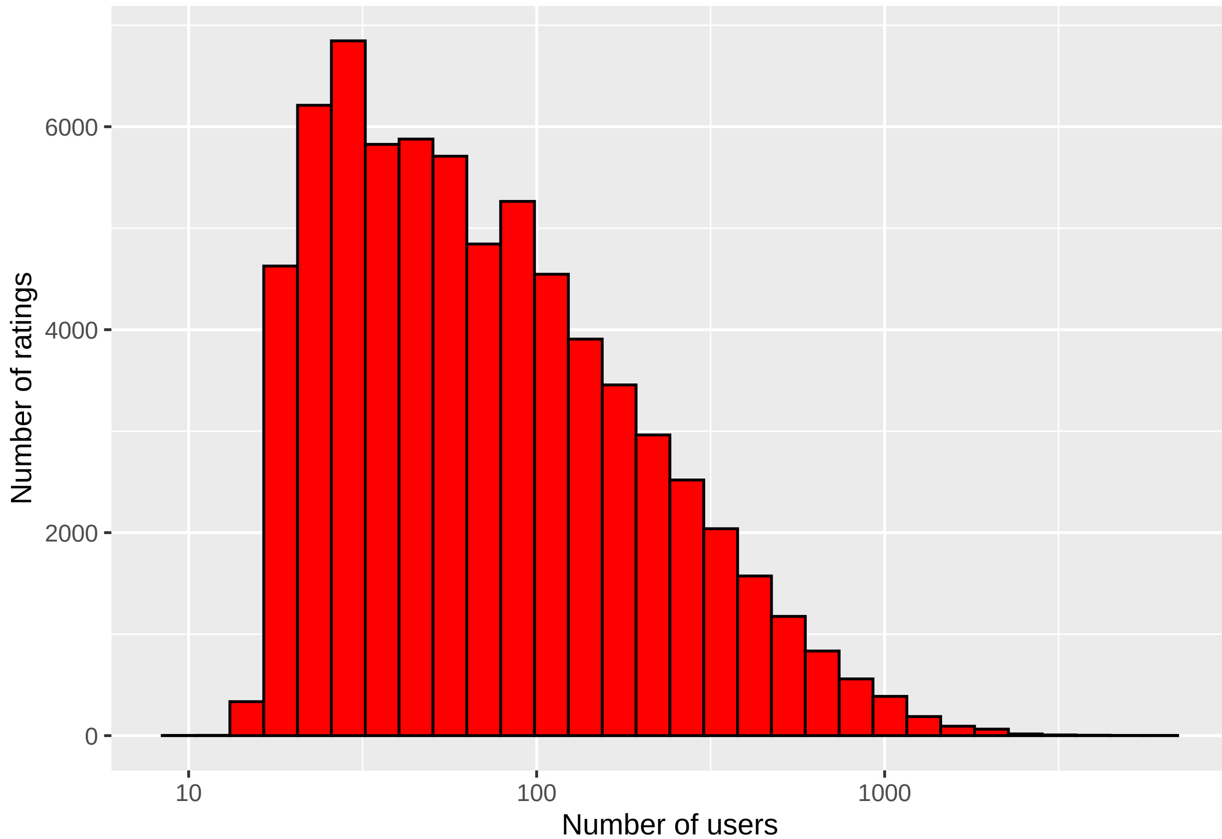
average rating across all movies

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

distribution of movie ratings



distribution of user ratings



Residual Mean Square Error formula for testing accuracy of predictions

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

Building the Recommendation System

Average movie rating model

average of all ratings across all users

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

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

predict all unknown ratings with mu

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

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

create a table to store results of prediction approaches

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

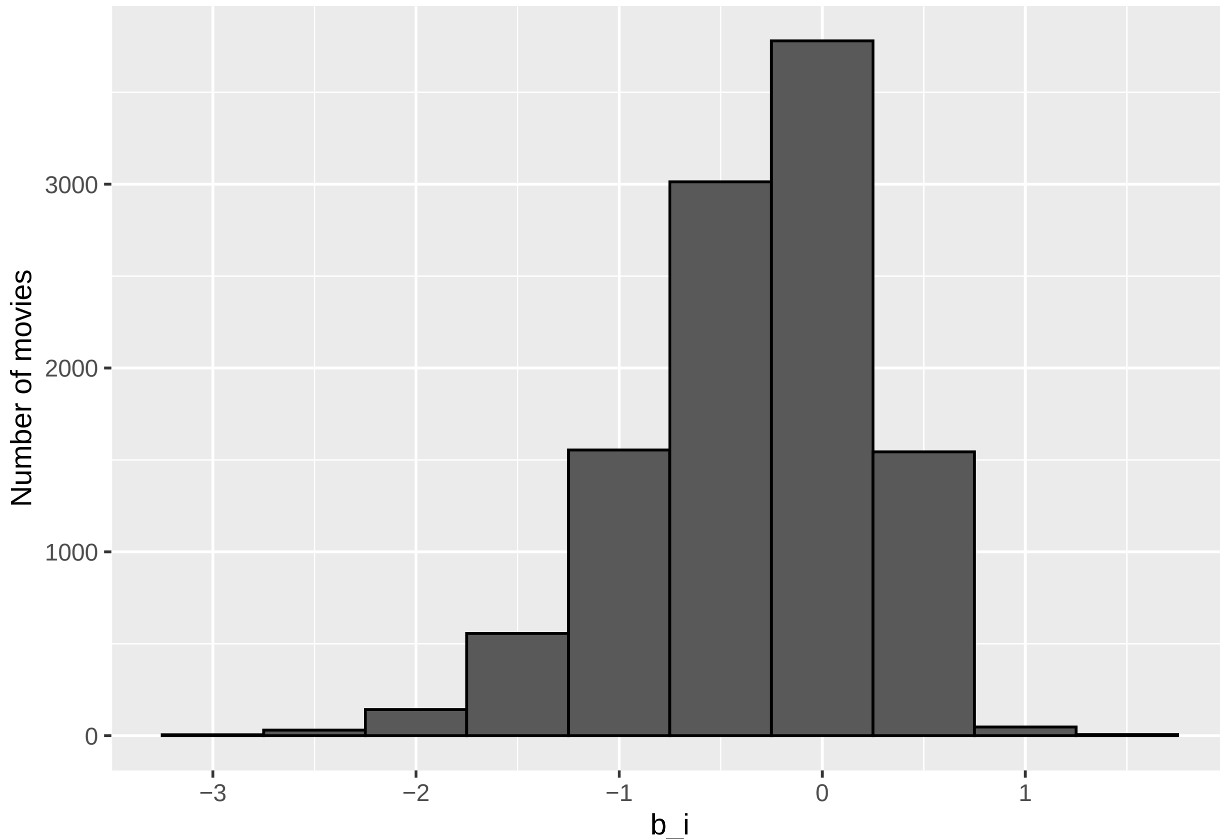
table showing naive RMSE prediction average

method	RMSE
Just the average	1.061202

model accounting for movie effect (b_i)

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

plot the number of movies with computed b__i



test and save RMSE results

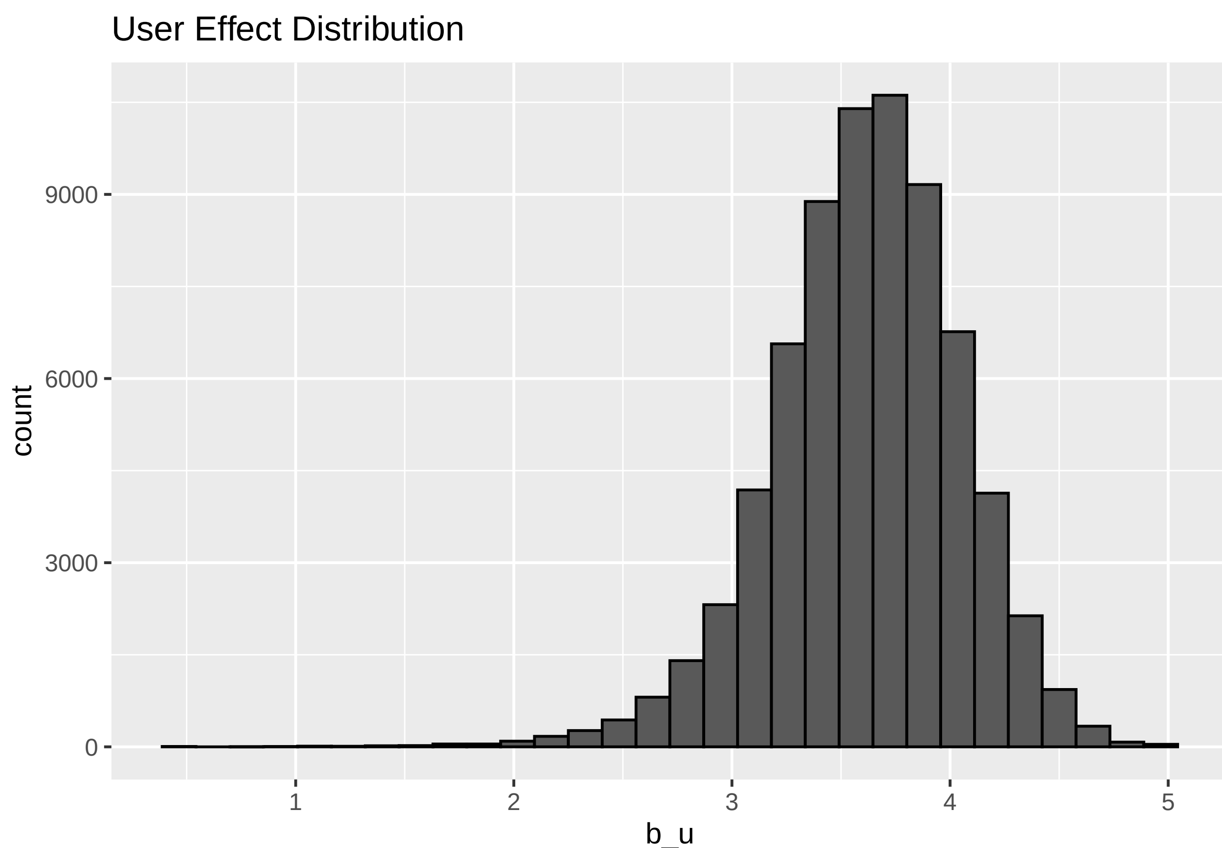
```
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i
model_1_rmse <- RMSE(predicted_ratings, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE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

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

table showing movie effect model results

method	RMSE
Just the average	1.0612018
Movie Effect Model	0.9439087

plot showing user effect for users with more than 100 ratings



model accounting for user effect (b__u) + movie effect

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

test and save new RMSE results

```
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,
                          data_frame(method = "Movie + User Effects Model",
                                     RMSE = model_2_rmse))
```

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

Regularization of movie + user effect model

lambda is a tuning parameter, chosen by cross-validation

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

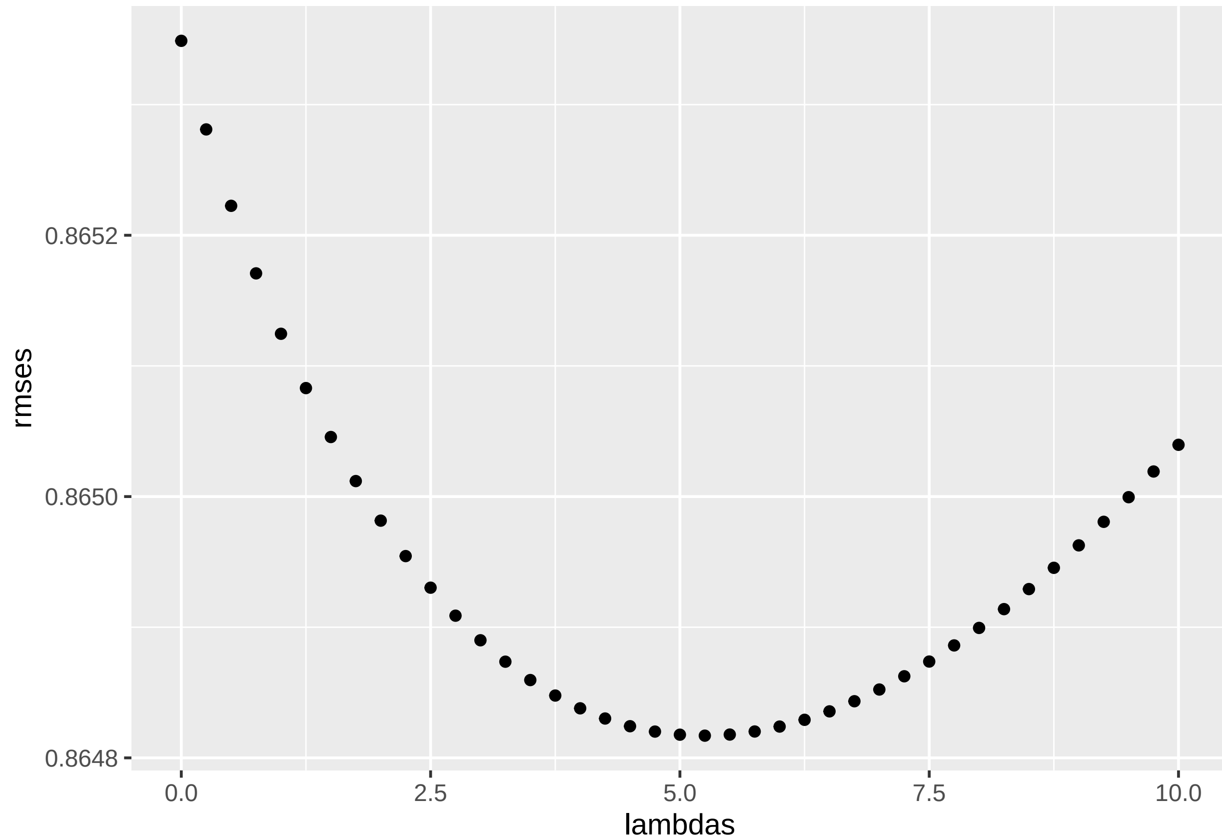
```


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



find optimal lambda

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

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

regularized model accounting for movie + user effect

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

table showing regularized movie + user effect model results

method	RMSE
Just the average	1.0612018

method	RMSE
Movie Effect Model	0.9439087
Movie + User Effects Model	0.8653488
Regularized Movie + User Effect Model	0.8648170

Matrix factorization of genres to refine prediction

split movies with multiple genres in 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
```

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) %>%
```

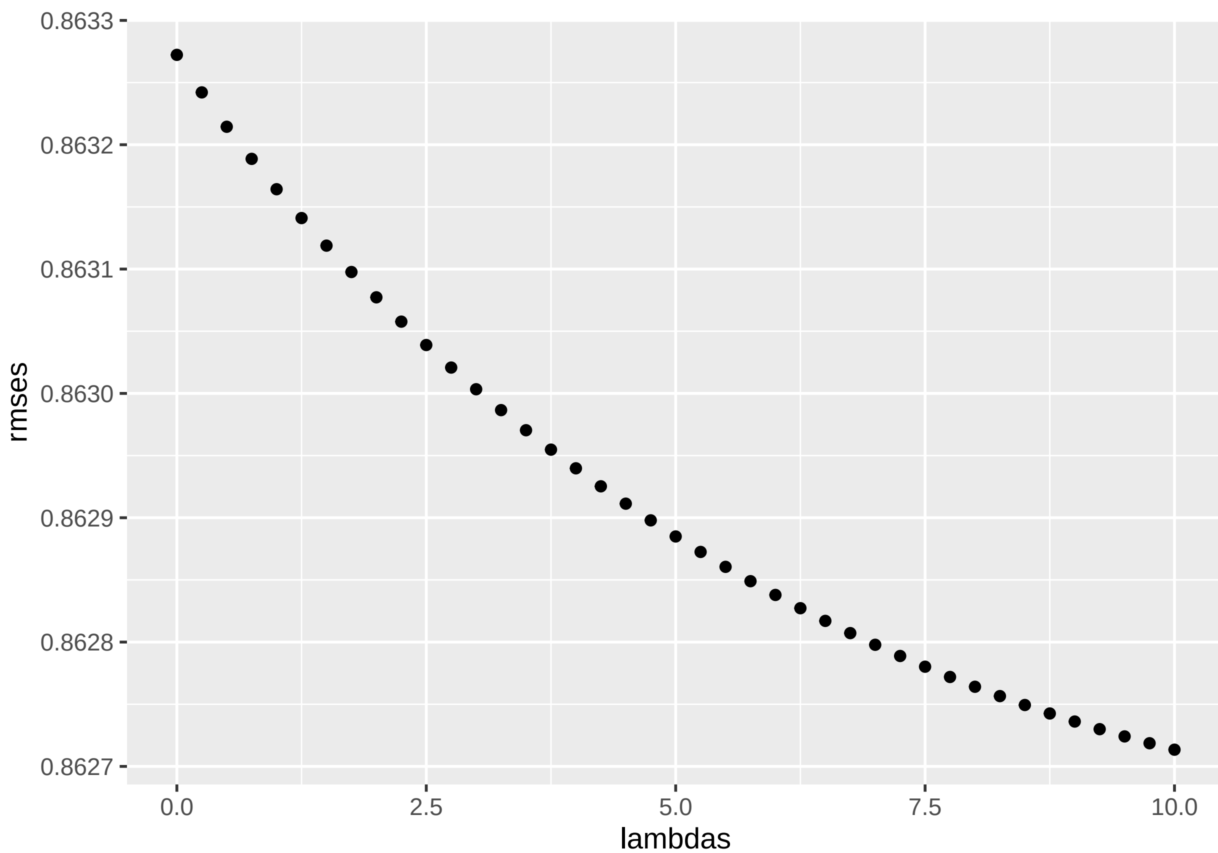
```
    summarise(b_g = sum(rating - b_i - b_u - mu)/(n()+1))

  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



find optimal lambda

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

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

regularized model accounting for movie + user + genre effect

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

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