Section-9-Movielens.R

tetra

2022-09-05

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

# Note: this process could take a couple of minutes

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

```
## Loading required package: tidyverse
```

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

```
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- 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
```

```
## 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
```

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

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

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

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

```
## Loading required package: tinytex
```

```
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("https://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 3.6 or earlier:
#movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
#                                           title = as.character(title),
#                                           genres = as.character(genres))
# if using R 4.0 or later:
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
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
```

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

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
```

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

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

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

# explore the data
# dimensions
dim(edx)
```

```
## [1] 9000055       6
```

```
# basic look at ratings
edx %>% filter(rating == 0) %>% tally()
```

```
##   n
## 1 0
```

```
edx %>% filter(rating == 3) %>% tally()
```

```
##         n
## 1 2121240
```

```
# number of unique movies
n_distinct(edx$movieId)
```

```
## [1] 10677
```

```
# number of unique users
n_distinct(edx$userId)
```

```
## [1] 69878
```

```
# number of movie ratings in a sample of common genres
genres = c("Drama", "Comedy", "Thriller", "Romance")
sapply(genres, function(g) {
  sum(str_detect(edx$genres, g))
})
```

```
##    Drama   Comedy Thriller  Romance
##  3910127  3540930  2325899  1712100
```

```
# number of ratings by titles
edx %>% group_by(movieId, title) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## `summarise()` has grouped output by 'movieId'. You can override using the
## `.groups` argument.
```

```
## # 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
```

```
# most given ratings
edx %>% group_by(rating) %>% summarize(count = n()) %>% top_n(5) %>%
  arrange(desc(count))
```

```
## Selecting by count
```

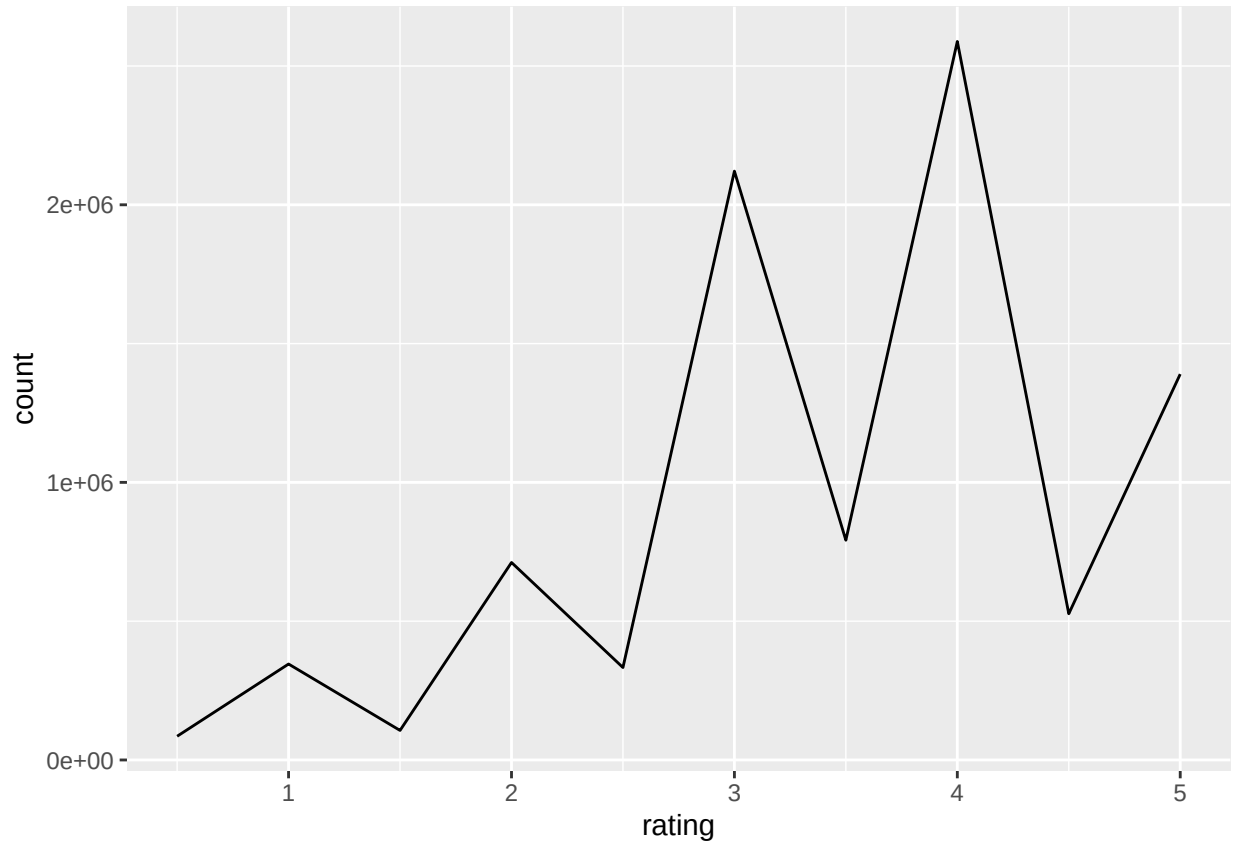
```
## # A tibble: 5 x 2
##   rating   count
##    <dbl>   <int>
## 1    4   2588430
## 2    3   2121240
## 3    5   1390114
## 4    3.5  791624
## 5    2    711422
```

```
# preference for full star ratings?
# counts of each given rating
edx %>% group_by(rating) %>% summarize(count = n())
```

```
## # A tibble: 10 x 2
##    rating   count
##     <dbl>   <int>
##  1    0.5   85374
##  2    1    345679
```

```
##  3    1.5  106426
##  4    2    711422
##  5    2.5  333010
##  6    3   2121240
##  7    3.5  791624
##  8    4   2588430
##  9    4.5  526736
## 10    5   1390114
```

```
# visualization of preference for full star ratings
edx %>%
  group_by(rating) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = rating, y = count)) +
  geom_line()
```



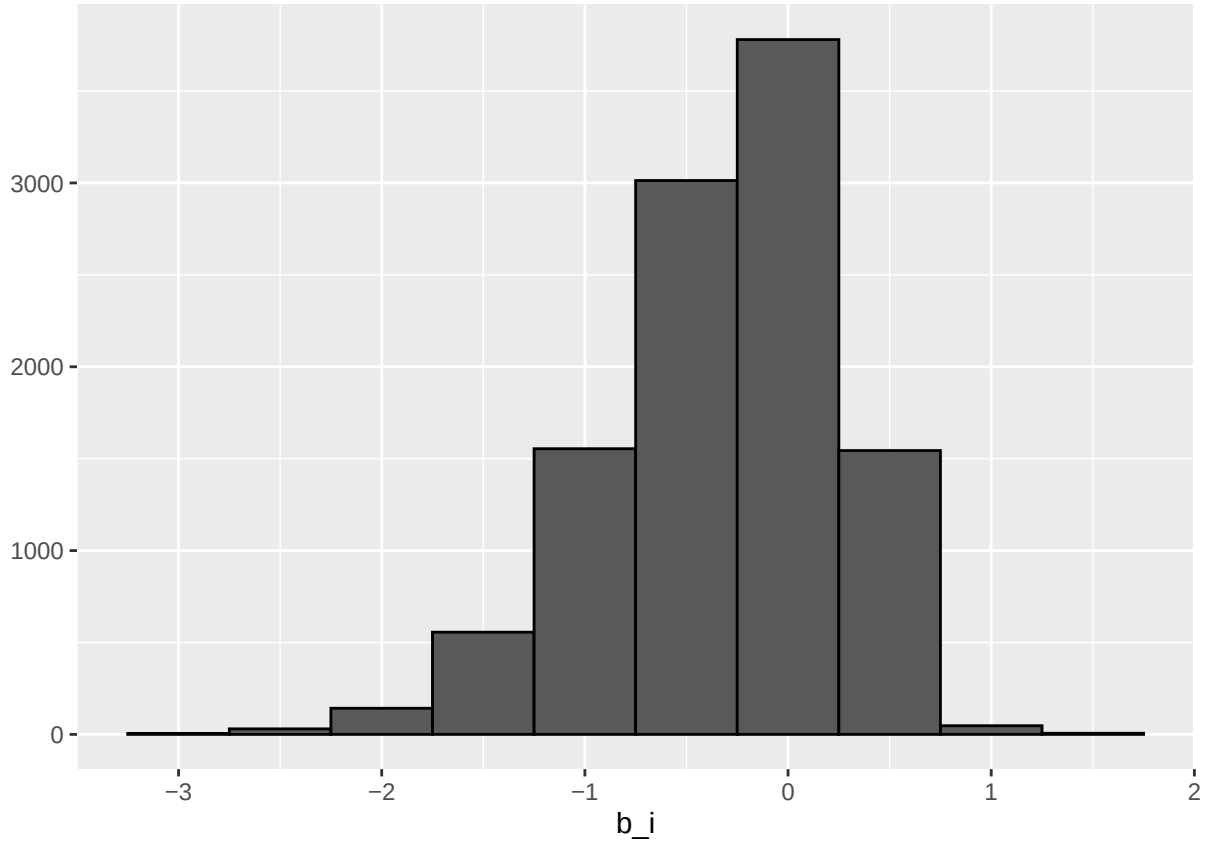

```
# create RMSE funciton to evaluate model results
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
# find average rating in edx set
edx_mean <- mean(edx$rating)
# print the average rating from the edx test est
edx_mean
```

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

```
# RMSE of guessing average rating against train set
naive_rmse <- RMSE(edx$rating, edx_mean)
# We see that this error is fairly large, more than 1.
naive_rmse
```

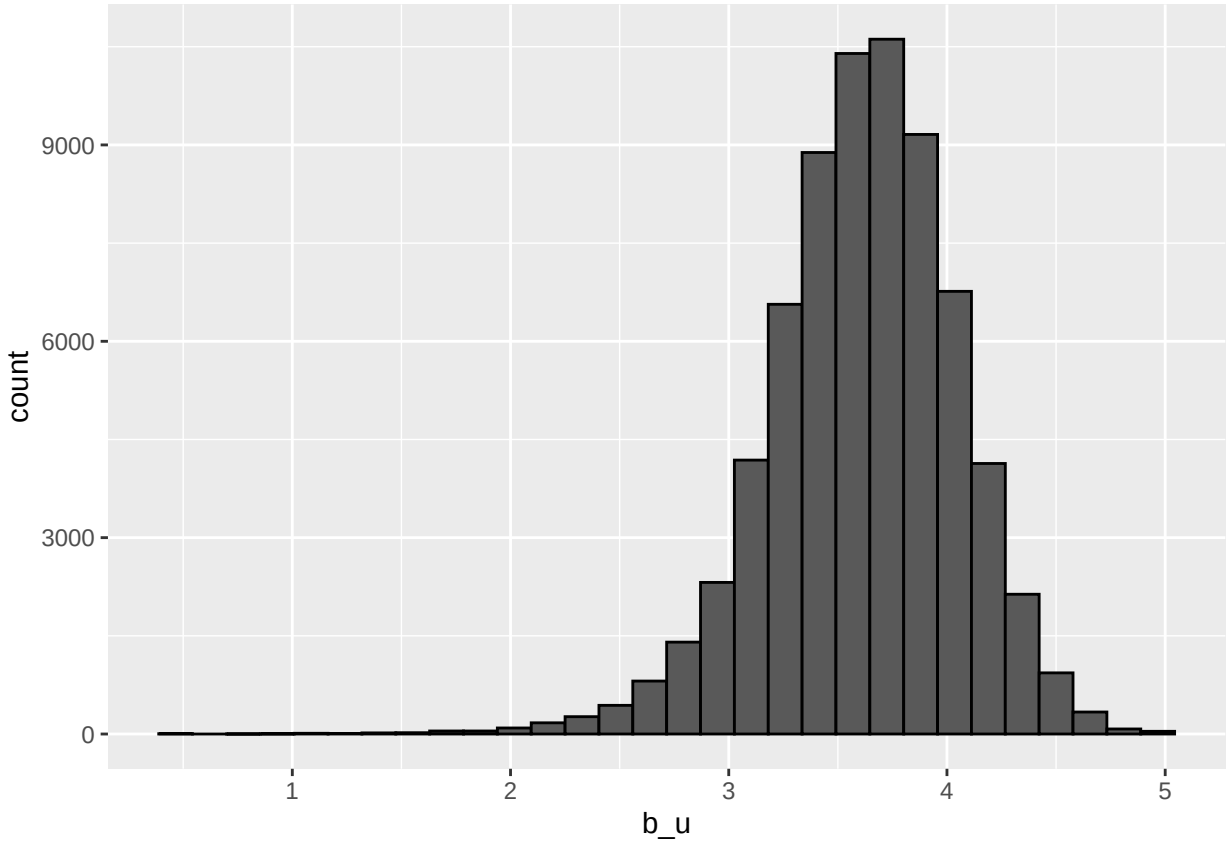
```
## [1] 1.060331
```

```
# add naive_rmse to a table of rmse results
rmse_results <- tibble(method = "Just the average", RMSE = naive_rmse)
# calculate a movie factor, average difference between a movies score and the overall average score
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - edx_mean))
# histogram of movie factors
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"))
```



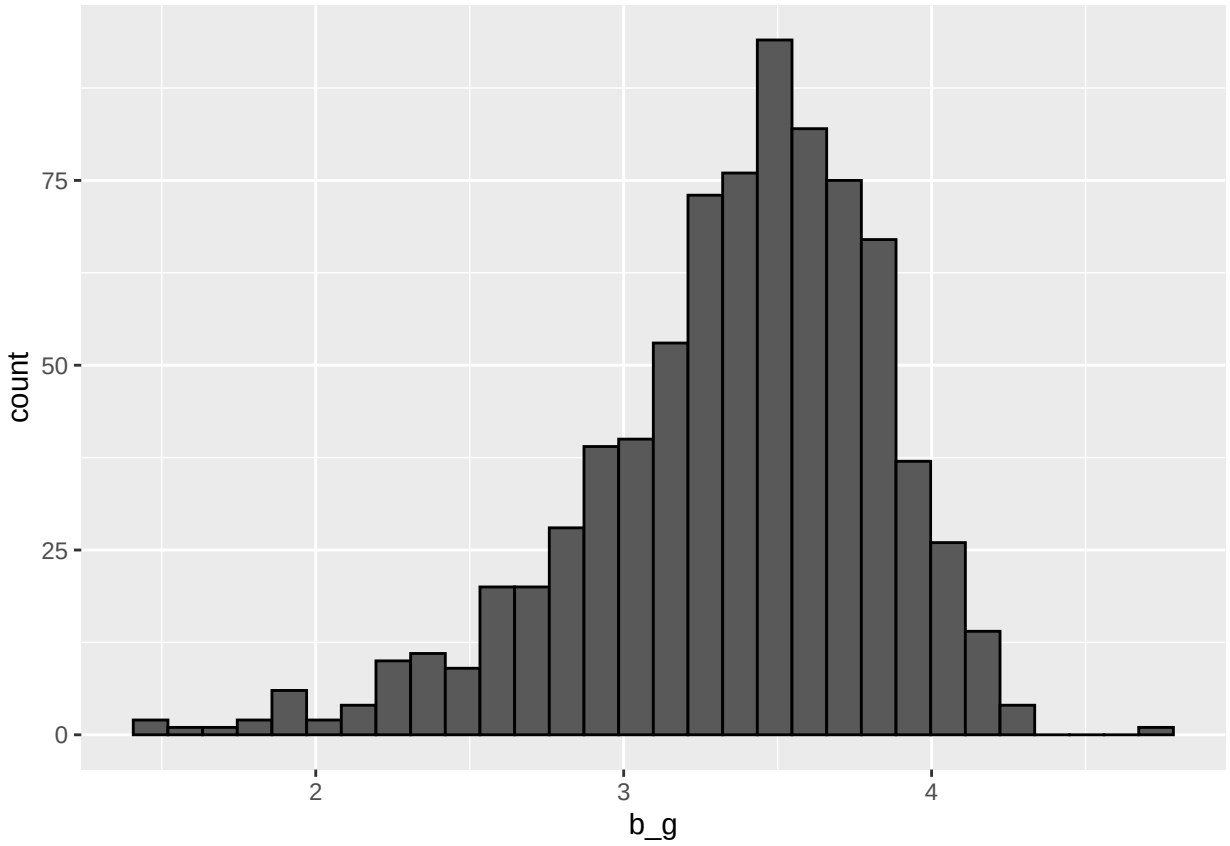

```
# prediction based on per-movie average combined with overall average
predicted_ratings_MEM <- edx_mean + edx %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i
# RMSE of new predictions
model_1_rmse <- RMSE(predicted_ratings_MEM, edx$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie Effect Model",
```

```
                                    RMSE = model_1_rmse ))
# plot of mean ratings by user
edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```




```
# calculate a user factor, average difference between a user's score and overall average including movi
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - edx_mean - b_i))
# predictions based on user factor and movie factor
predicted_ratings_UEM <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = edx_mean + b_i + b_u) %>%
  .$pred
# RMSE of model 2
model_2_rmse <- RMSE(predicted_ratings_UEM, edx$rating)
# added to results table
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie + User Effects Model",
                                     RMSE = model_2_rmse ))
```

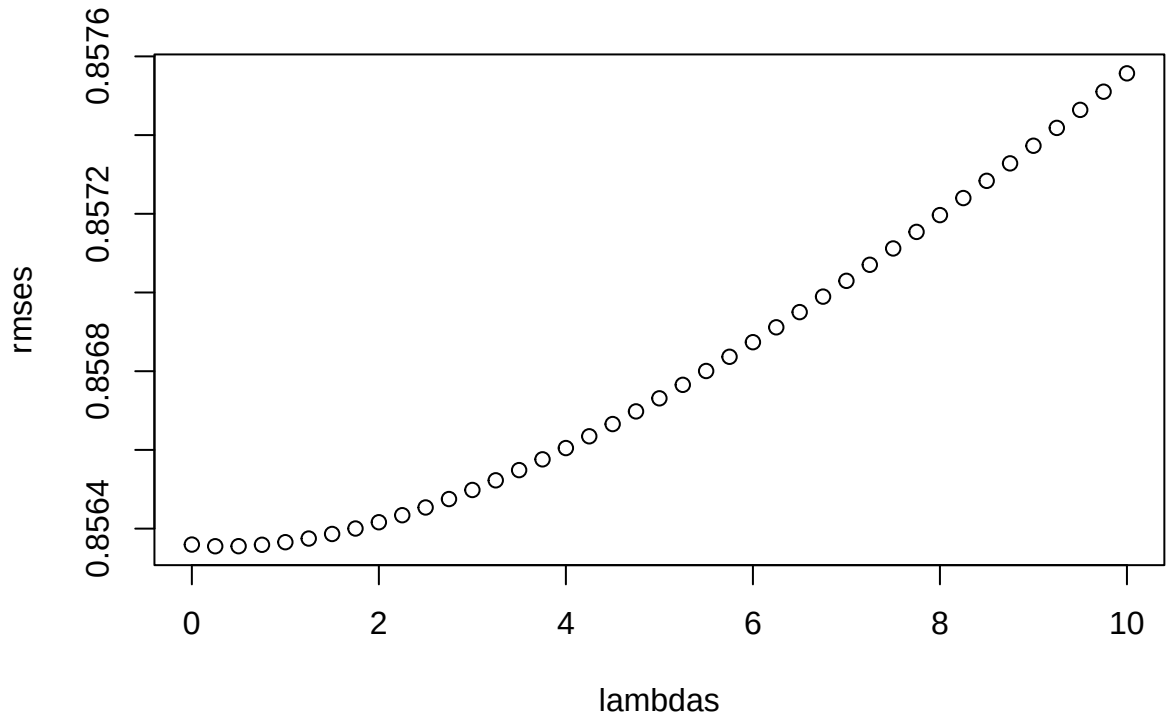
```r
# plot of genre effects, mean ratings by genre
edx %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating)) %>%
  ggplot(aes(b_g)) +
  geom_histogram(bins = 30, color = "black")
```




```r
# calculate genre factor, average difference between genre's score and overall average include u and i
genre_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - edx_mean - b_i - b_u))
# predictions based on genre, user, and movie effect factors
predicted_ratings_GEM <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genre_avgs, by='genres') %>%
  mutate(pred = edx_mean + b_i + b_u + b_g) %>%
  .$pred
# RMSE of model 3
model_3_rmse <- RMSE(predicted_ratings_GEM, edx$rating)
# added to results table
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie + User + Genre Effects Model",
```

```
                                    RMSE = model_3_rmse ))
# test lambdas for regularization and apply to the model
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  b_g <- edx %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - b_i - b_u - mu)/(n()+l))
  predicted_ratings <-
    edx %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    mutate(pred = mu + b_i + b_u + b_g) %>%
    .$pred
  return(RMSE(predicted_ratings, edx$rating))
})

# plot to visualize effect of lambda on RMSE
plot(lambdas,rmses)
```




```
# store best lambda and add newest RMSE to table of results
lambda <- lambdas[which.min(rmses)]
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Regularized Movie + User + Genre Effect Model",
                                 RMSE = min(rmses)))
# Check progression of RMSE as models evolve
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0603313
Movie Effect Model	0.9423475
Movie + User Effects Model	0.8567039
Movie + User + Genre Effects Model	0.8563595
Regularized Movie + User + Genre Effect Model	0.8563552

```
# add the regularization factor into the models for movie, user, and genre
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - edx_mean)/(n()+lambda))
user_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - edx_mean - b_i)/(n()+lambda))
genre_avgs <- edx %>%
```

```
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(genres) %>%
  summarize(b_g = sum(rating - edx_mean - b_i - b_u)/(n()+lambda))
# predict validation set with all 3 factors now including lambda
y_hat <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genre_avgs, by='genres') %>%
  mutate(pred = edx_mean + b_i + b_u + b_g) %>%
  .$pred
# calculate RMSE of prediction against validation set
RMSE(validation$rating,y_hat)
```

```
## [1] 0.8648817
```

```
# add target final results to table
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Target Final Reslut",
                                 RMSE = 0.8649000))
# add final results to table
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Final Model vs Validation RMSE",
                                     RMSE = RMSE(validation$rating,y_hat)))
# final RMSE results table
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0603313
Movie Effect Model	0.9423475
Movie + User Effects Model	0.8567039
Movie + User + Genre Effects Model	0.8563595
Regularized Movie + User + Genre Effect Model	0.8563552
Target Final Reslut	0.8649000
Final Model vs Validation RMSE	0.8648817