MovieLens Edx Project

Andrew White

8/5/2019

Introduction

The MovieLens 10M dataset is a dataset of 10 million movie reviews which I will use to train and create a movie recommendation system using machine learning. The goal of this project is to create a movie recommendation system with a root mean squared error of less than 0.8775. I will start with some data exploration to get a better understanding of the dataset. This data exploration is essential for finding trends in the data which can be exploited to create and improve prediction models. I will use an approach called matrix factorization which uses different effects (or biases as Netflix calls them) to predict which movies are closely associated. After matrix factorization, I will use regularization to try to improve those models.

```
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")
```

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")
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[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")
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)
```

Data Exploration

Check out the size of the dataset and change the options to show up to 7 digits

```
options(digits = 7)
dim(edx)
```

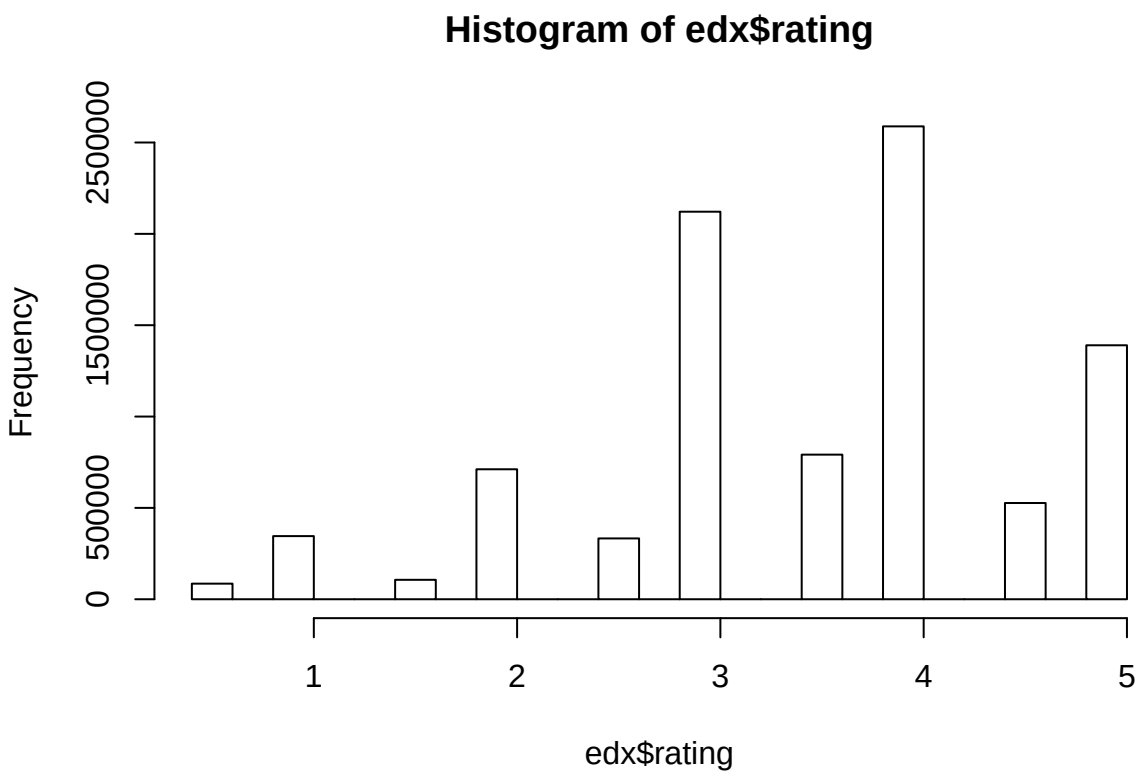
```
## [1] 9000055       6
```

Look at the ratings

```
length(edx$rating)
```

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

```
hist(edx$rating)
```



```
edx%>% group_by(rating)%>%
  summarize(count=n())%>%
  arrange(desc(count))%>%
  head()
```

```
## # A tibble: 6 x 2
##   rating   count
##    <dbl>   <int>
## 1    4   2588430
## 2    3   2121240
## 3    5   1390114
## 4    3.5  791624
## 5    2    711422
## 6    4.5  526736
```

```
mean(edx$rating)
```

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

How many movies and users?

```
length(unique(edx$title))
```

```
## [1] 10676
```

```
length(unique(edx$userId))
```

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

What is the breakdown by genre?

```
edx %>% separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 20 x 2
##    genres               count
##    <chr>                <int>
##  1 Drama              3910127
##  2 Comedy             3540930
##  3 Action             2560545
##  4 Thriller           2325899
##  5 Adventure          1908892
##  6 Romance            1712100
##  7 Sci-Fi             1341183
##  8 Crime              1327715
##  9 Fantasy             925637
## 10 Children            737994
## 11 Horror              691485
```

```
## 12 Mystery             568332
## 13 War                 511147
## 14 Animation           467168
## 15 Musical             433080
## 16 Western             189394
## 17 Film-Noir           118541
## 18 Documentary          93066
## 19 IMAX                  8181
## 20 (no genres listed)       7
```

Look at the most rated movies

```
edx %>% group_by(movieId, title) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # 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) (19~ 25672
## 10     150 Apollo 13 (1995)                                          24284
## # ... with 10,667 more rows
```

Method/Analysis

Create Test/Training sets within EDX data

```
set.seed(1)
test_index <- createDataPartition(y = edx$rating, times = 1,
                                  p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

Create RMSE function to measure quality of the prediction

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

For the first prediction, I can just use the average as our baseline to improve upon

```
mu_hat <- mean(train_set$rating)
mu_hat
```

```
## [1] 3.512482
```

```
naive_rmse <- RMSE(test_set$rating, mu_hat)
naive_rmse
```

```
## [1] 1.059904
```

```
predictions <- rep(2.5, nrow(test_set))
RMSE(test_set$rating, predictions)
```

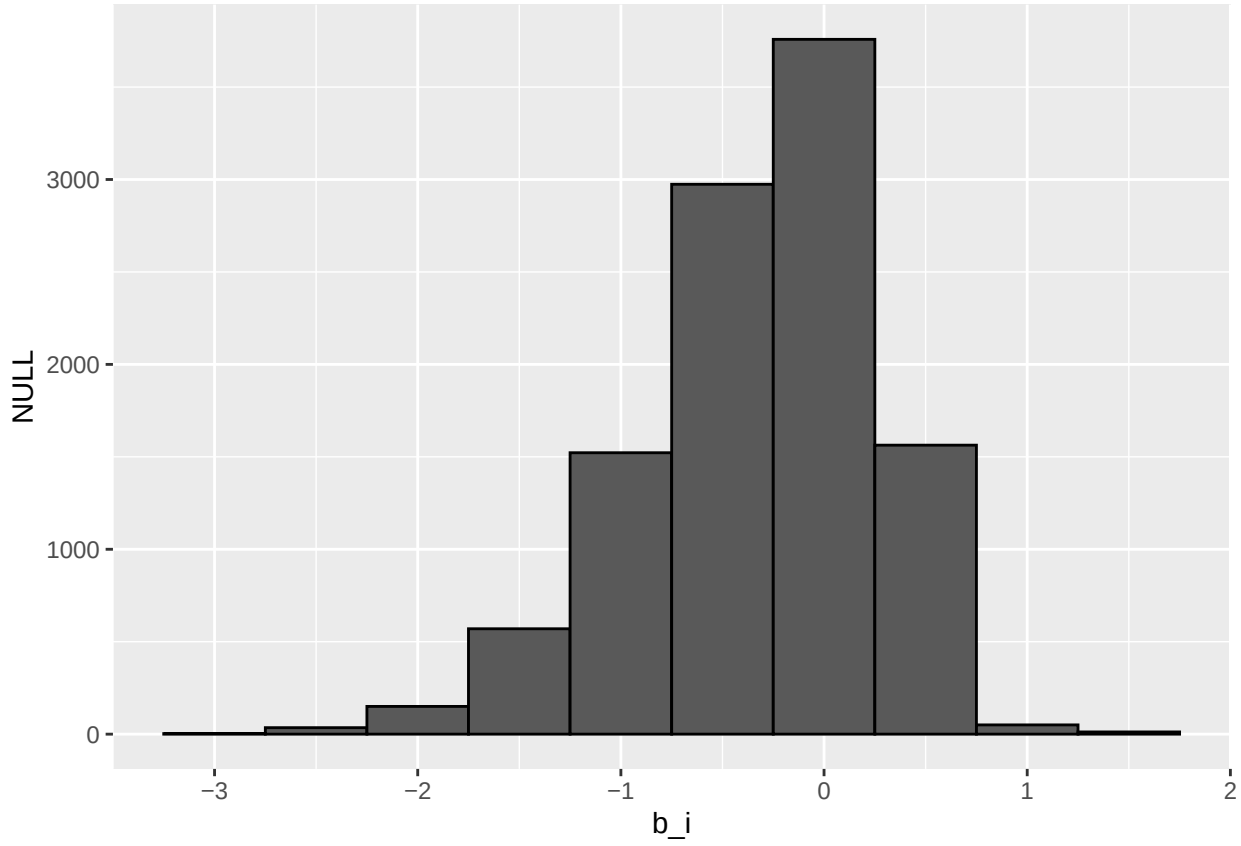
```
## [1] 1.465736
```

Create a table named rmse_results to store results so I can compare RMSE results across methods

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

I see the naive rmse is 1.05990. I can improve this by taking into account some general trends or effects in the data that impact rating. I saw in the movie rating data that there is an effect (or bias as Netflix calls it) caused by some movies having higher average ratings Call this b_i and adjust the recommendation system to account for this.

```
mu <- mean(train_set$rating)
movie_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"))
```



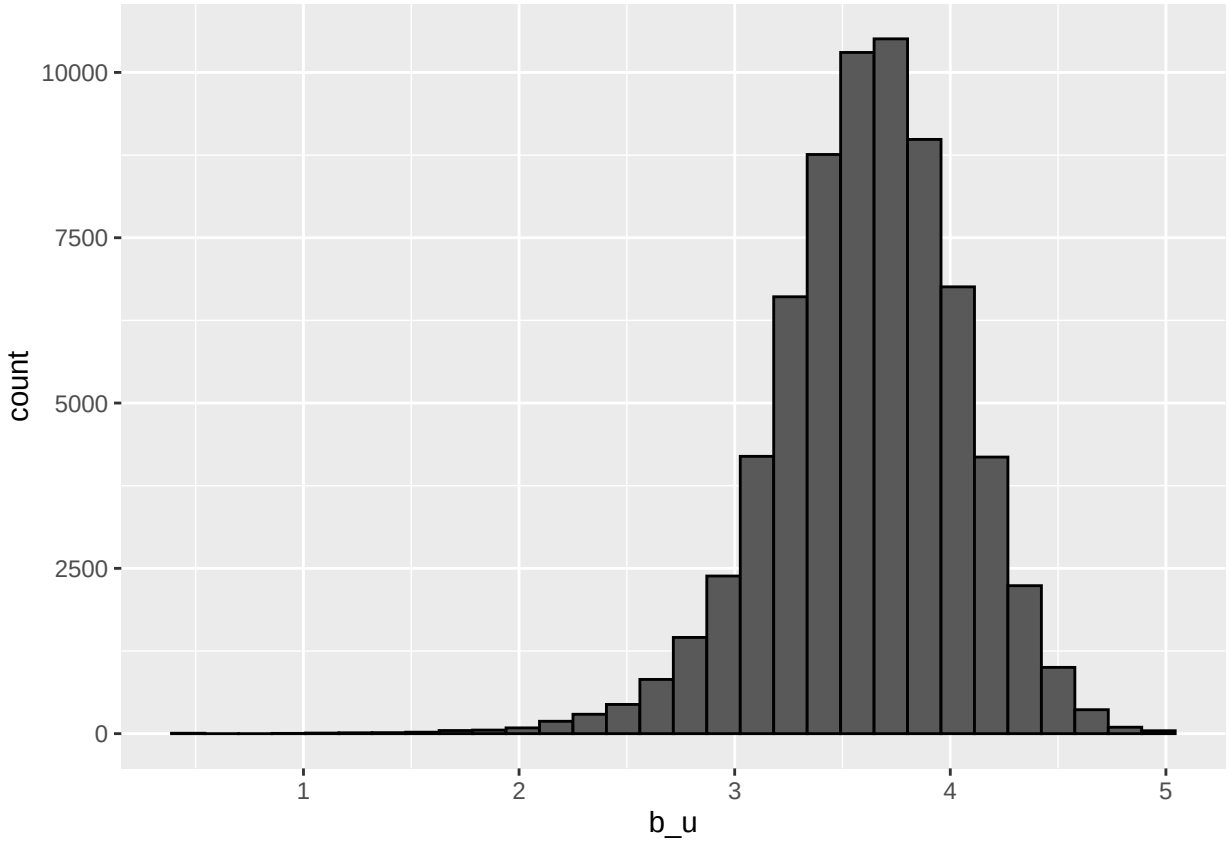

```
predicted_ratings <- mu + test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  .$b_i

model_1_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie Effect Model",
                                     RMSE = model_1_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0599043
Movie Effect Model	0.9437429

I see adding the movie effect lowers the RMSE to .94374 which is a nice improvement, but it is still not low enough to be helpful. As I saw in the user data, there is a trend that some users are harsher than others when it comes to ratings. I will call this trend the user effect (b_u) and use it to try to further reduce the RMSE

```
train_set %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
```

```
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```




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

Next I combine both effects to improve the model by redefining predicted_ratings with movie and user effects

```
predicted_ratings <- test_set %>%
  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, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model",
                                     RMSE = model_2_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0599043
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8431355

It dropped all the way down to 0.84313! This is great, but by testing a few other techniques, I might be able to improve the model further. First look at the 10 biggest adjustments are at the top and bottom.

Make a movie_titles dataframe so it is easier to see the effects and titles

```
movie_titles <- edx %>%
  select(movieId, title) %>%
  distinct()
```

Top 10

```
train_set %>% dplyr::count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	b_i	n
Hellhounds on My Trail (1999)	1.487518	1
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.487518	3
Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	1.487518	2
Shadows of Forgotten Ancestors (1964)	1.487518	1
Money (Argent, L') (1983)	1.487518	1
Fighting Elegy (Kenka erejii) (1966)	1.487518	1
Sun Alley (Sonnenallee) (1999)	1.487518	1
Aerial, The (La Antena) (2007)	1.487518	1
Blue Light, The (Das Blaue Licht) (1932)	1.487518	1
More (1998)	1.404184	6

Bottom 10

```
train_set %>% dplyr::count(movieId) %>%
  left_join(movie_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

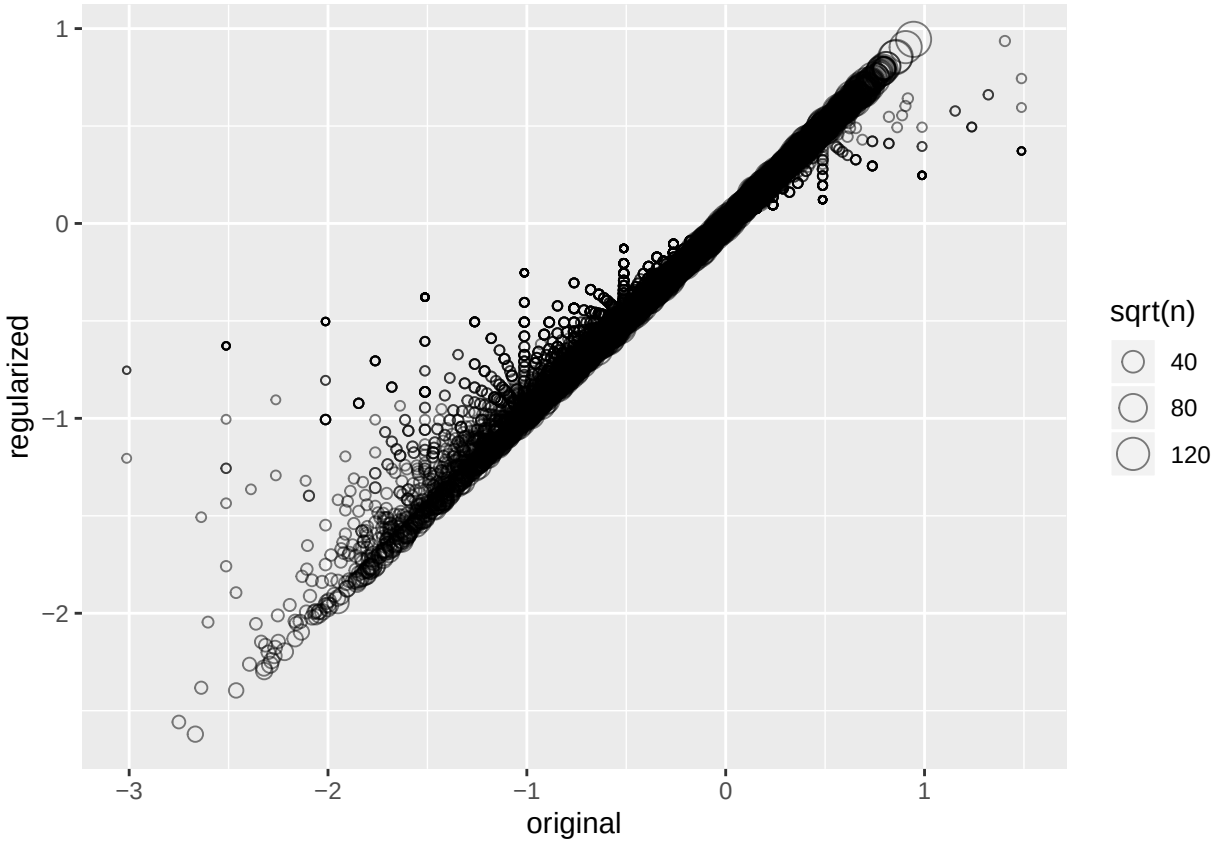
title	b_i	n
Besotted (2001)	-3.012482	1
Confessions of a Superhero (2007)	-3.012482	1

title	b_i	n
War of the Worlds 2: The Next Wave (2008)	-3.012482	2
SuperBabies: Baby Geniuses 2 (2004)	-2.749982	40
From Justin to Kelly (2003)	-2.667244	168
Legion of the Dead (2000)	-2.637482	4
Disaster Movie (2008)	-2.637482	28
Hip Hop Witch, Da (2000)	-2.603391	11
Criminals (1996)	-2.512482	1
Mountain Eagle, The (1926)	-2.512482	1

They are barely viewed movies so lets use regularization to account for the number of reviews

Use a random lambda 3 for now that I can adjust later

```
lambda_i<- 3
mu <- mean(train_set$rating)
movie_reg_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda_i), n_i = n())

data_frame(original = movie_avgs$b_i,
           regularized = movie_reg_avgs$b_i,
           n = movie_reg_avgs$n_i) %>%
  ggplot(aes(original, regularized, size=sqrt(n))) +
  geom_point(shape=1, alpha=0.5)
```



Review them after regularizaton

Top 10

```
train_set %>%
  dplyr::count(movieId) %>%
  left_join(movie_reg_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	b_i	n
Shawshank Redemption, The (1994)	0.9447090	22363
More (1998)	0.9361230	6
Godfather, The (1972)	0.9048132	14107
Usual Suspects, The (1995)	0.8567890	17315
Schindler's List (1993)	0.8514402	18567
Rear Window (1954)	0.8086363	6325
Casablanca (1942)	0.8045763	9027
Double Indemnity (1944)	0.7970496	1711
Third Man, The (1949)	0.7960352	2420
Dark Knight, The (2008)	0.7957351	1900

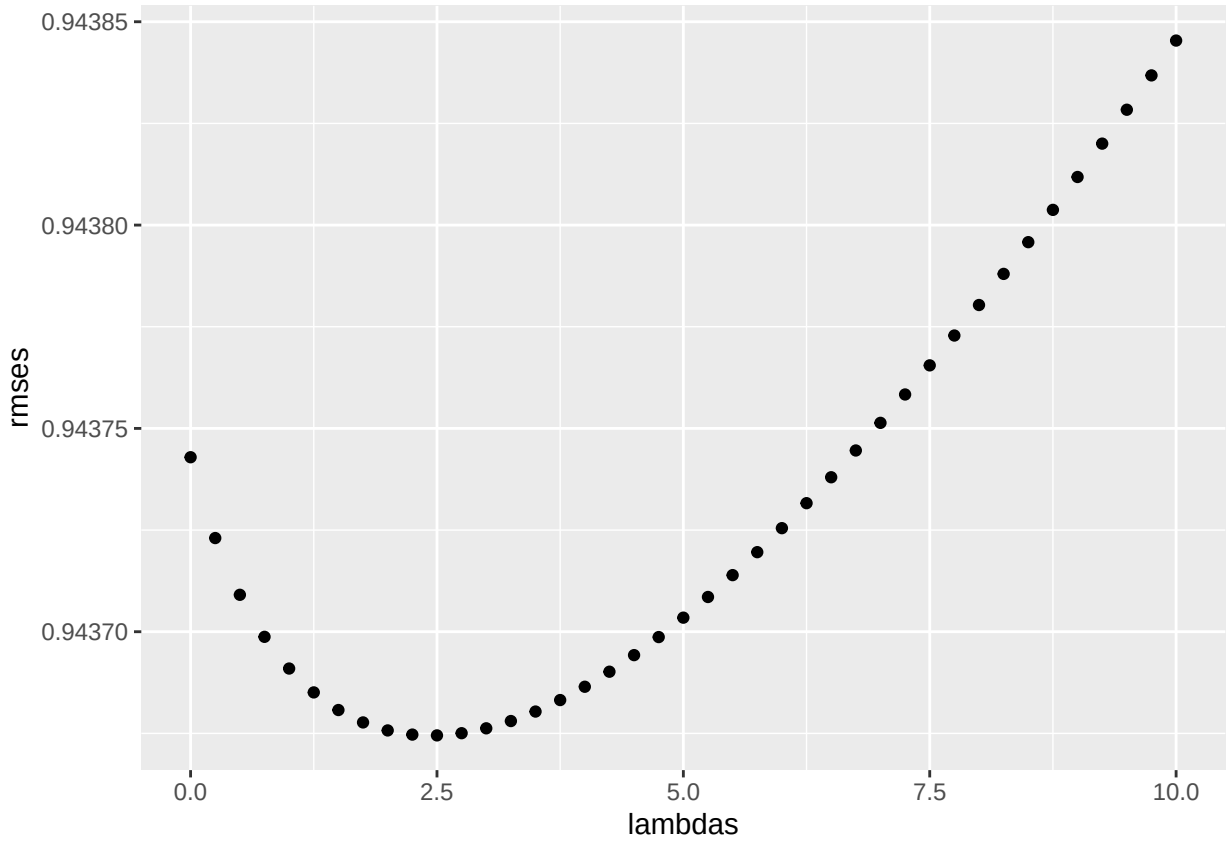
Bottom 10

```
train_set %>%
  dplyr::count(movieId) %>%
  left_join(movie_reg_avgs) %>%
  left_join(movie_titles, by="movieId") %>%
  arrange(b_i) %>%
  select(title, b_i, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	b_i	n
From Justin to Kelly (2003)	-2.620450	168
SuperBabies: Baby Geniuses 2 (2004)	-2.558123	40
PokÃ©mon Heroes (2003)	-2.396076	109
Disaster Movie (2008)	-2.382242	28
Glitter (2001)	-2.296561	282
Barney's Great Adventure (1998)	-2.281269	168
Gigli (2003)	-2.264318	249
Carnosaur 3: Primal Species (1996)	-2.261789	51
Pokemon 4 Ever (a.k.a. PokÃ©mon 4: The Movie) (2002)	-2.243257	168
Son of the Mask (2005)	-2.218303	128

Now the movies with the biggest adjustments have very many ratings. That improved a lot, but lets tune the lambda to get the best regularization

```
lambdas <- seq(0, 10, 0.25)
mu <- mean(train_set$rating)
just_the_sum <- train_set %>%
  group_by(movieId) %>%
  summarize(s = sum(rating - mu), n_i = n())
rmses <- sapply(lambdas, function(l){
  predicted_ratings <- test_set %>%
    left_join(just_the_sum, by='movieId') %>%
    mutate(b_i = s/(n_i+l)) %>%
    mutate(pred = mu + b_i) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})
qplot(lambdas, rmses)
```




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

```
## [1] 2.5
```

I see 2.5 is the best lambda for RMSE so I will redefine movie_reg_avgs with lambda_i=2.25

```
lambda_i<-2.5
movie_reg_avgs <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+lambda_i), n_i = n())
```

Now check the RMSE of the regularized Movie Effect

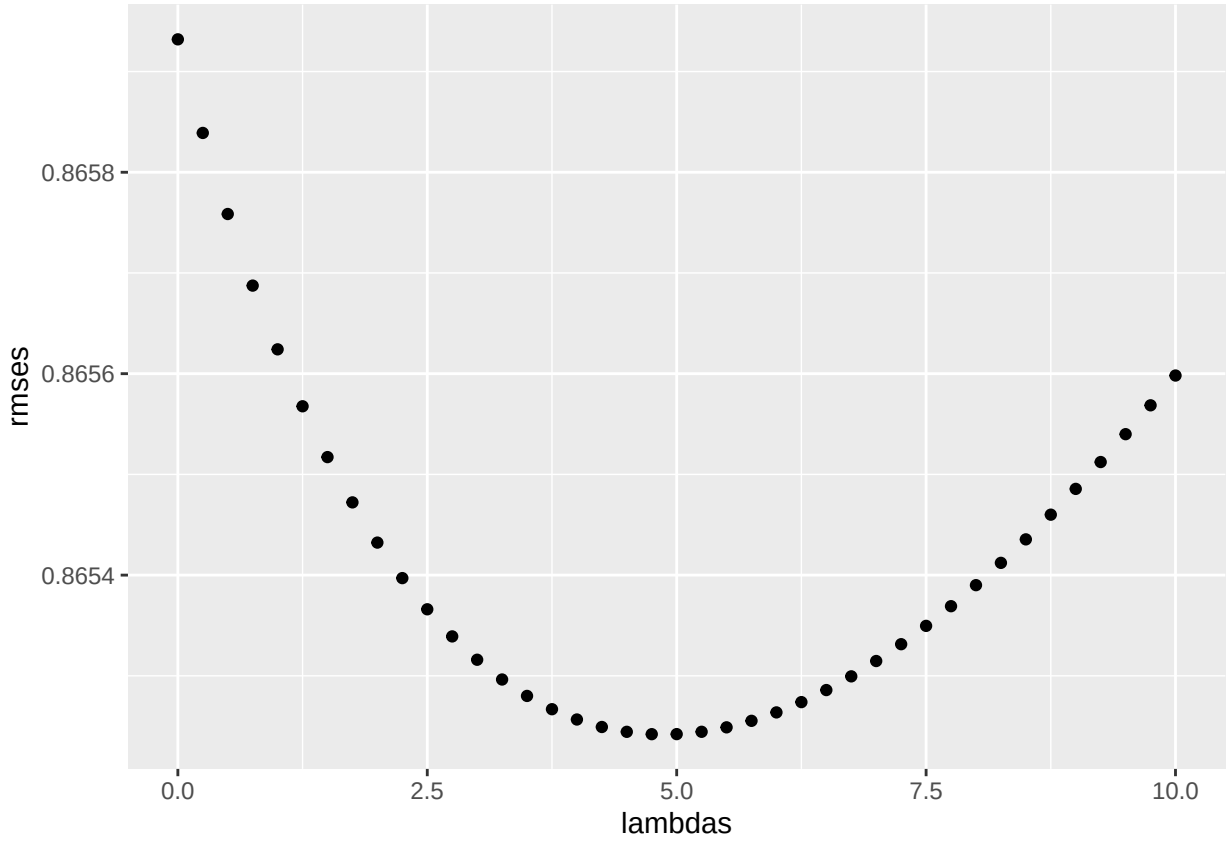
```
predicted_ratings <- test_set %>%
  left_join(movie_reg_avgs, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  .$pred

model_3_rmse <- RMSE(predicted_ratings, test_set$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie Effect Model",
                                     RMSE = model_3_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0599043
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8431355
Regularized Movie Effect Model	0.9436745

It's 0.94367 which is only a tiny improvement over the unregularized model 1. Perhaps regularization makes a larger improvement to model 2.

I'll look for the optimal lambda for user effect and movie effect and use that to find the RMSE for the regularized combined user and movie effect

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(train_set$rating)
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- train_set %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <-
    test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})

qplot(lambdas, rmses)
```




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

```
## [1] 4.75
```

The best lambda for movie and user effects is 4.75 which gives use an RMSE of 0.865

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

method	RMSE
Just the average	1.0599043
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8431355
Regularized Movie Effect Model	0.9436745
Regularized Movie + User Effect Model	0.8652421

With that lambda I get a RMSE of 0.86524 which is much better than what I started with, but not an improvement over model 2

Results

```
rmse_results
```

```
## # A tibble: 5 x 2
##   method                                RMSE
##   <chr>                                <dbl>
## 1 Just the average                     1.06
## 2 Movie Effect Model                   0.944
## 3 Movie + User Effects Model           0.843
## 4 Regularized Movie Effect Model       0.944
## 5 Regularized Movie + User Effect Model 0.865
```

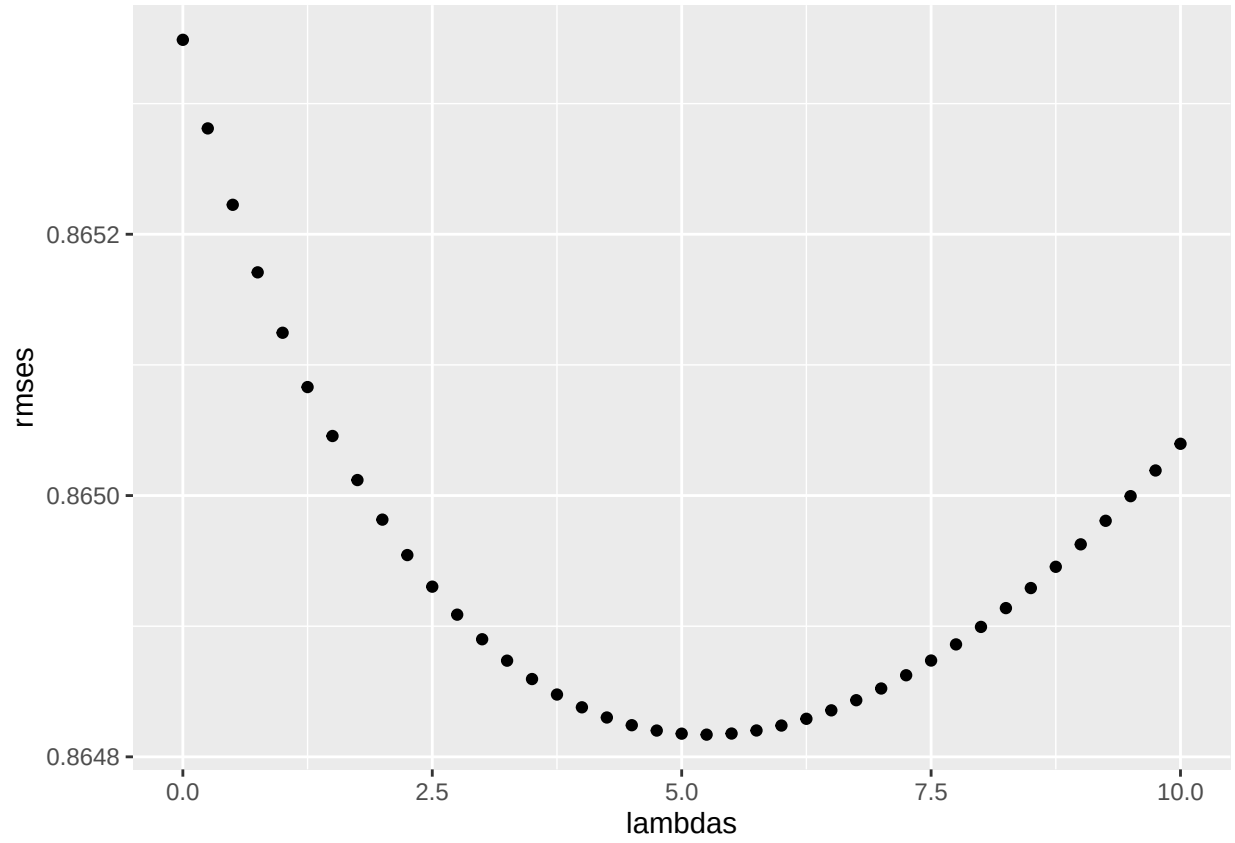
The combined user and movie effect dropped the RMSE all the way down to 0.84313 which was the best RMSE I achieved on the test set.

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

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

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

predicted_ratings_fin <- validation %>%
  left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>%
  mutate(pred = mu_fin + b_i + b_u) %>%
  .$pred

model_2_rmse_fin <- RMSE(predicted_ratings_fin, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie + User Effects Model_fin",
                                     RMSE = model_2_rmse_fin ))
rmse_results
```

```
## # A tibble: 6 x 2
##   method                                RMSE
##   <chr>                                <dbl>
## 1 Just the average                     1.06
## 2 Movie Effect Model                   0.944
## 3 Movie + User Effects Model           0.843
## 4 Regularized Movie Effect Model       0.944
## 5 Regularized Movie + User Effect Model 0.865
## 6 Movie + User Effects Model_fin       0.865
```

The resulting RMSE of 0.86534 is higher than expected. The RMSE was lower in the training data likely because of overtraining in the test data. The code for the final model using regularized user and movie effects is below.

```
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  return(RMSE(predicted_ratings, validation$rating))
})

qplot(lambdas, rmses)
```




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

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

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Regularized Movie + User Effect Model_fin",
                                     RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Just the average	1.0599043
Movie Effect Model	0.9437429
Movie + User Effects Model	0.8431355
Regularized Movie Effect Model	0.9436745
Regularized Movie + User Effect Model	0.8652421
Movie + User Effects Model_fin	0.8653488
Regularized Movie + User Effect Model_fin	0.8648170

Now the RMSE is 0.86481 which is even better.This improved the RMSE by 0.19508 which means that the movie recommendations will be 19.5% more accurate than simply guessing the average.

Conclusion

The final RMSE of 0.86481 is a significant improvement over the naive estimate I started with. However there is always room for improvement. For further study, I could look for other effects in the data such as genre and see how that improves the prediction. In future models, care must be taken to not overtrain the model as that will give a false sense of how strong the prediction model is. While regularization did not offer an improvement over in the training set, it did improve on the validation set. This final RMSE is below the 0.8649 threshold, which meets the Netflix challenge.