Predicting Movie Ratings

Andrew Ertell

10/21/2020

Executive Overview

This report details an approach for building an algorithm to predict movie ratings. An existing movie ratings data set from MovieLens is used to train a model that estimates ratings for an individual user/movie pairs. The model takes advantage of observable biases in the data and utilizes regularization to increase the accuracy of the predictions.

Dataset details

The publicly available MovieLens 10M Dataset contains 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users.

Validation / Goal

A subset of ratings (~10%) will be held back for validation of the final algorithm performance. Performance will be evaluated by calculating the Residual Mean Squared Error (RMSE) of our predicted ratings vs. the true ratings. The most accurate model is the one that minimizes RMSE.

Approach

- Explore the data for possible bias that could help improve predictions.
- Use regularization to minimize variability for small sample sizes (films/users with few ratings)

Analysis

Preparation

First, the MovieLens 10M Dataset is downloaded and prepared for analysis. Then the validation set is created. A semi_join is used on the validation set to make sure that all movieIds and userIds in the validation set are in the main dataset (named edx).

```
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/
```

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

if(!exists("movies") && !exists("ratings")) {
  f_movies <- "ml-10M100K/ratings.dat"
  f_ratings <- "ml-10M100K/movies.dat"
  if(!file.exists(f_movies) && !file.exists(f_ratings)) {
    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")
  } else {
    ratings <- fread(text = gsub("::", "\t", readLines("ml-10M100K/ratings.dat")),
                     col.names = c("userId", "movieId", "rating", "timestamp"))
    movies <- str_split_fixed(readLines("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))

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)` instead
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") %>%
  semi_join(edx, by = "genres")

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

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

#############################################################
# The validation set is only for the testing at the end.
#############################################################
```

The main dataset (edx) is also split into a training set and test set (~20% of the results) for testing the accuracy of the models as the algorithm is built.

```r
# set a seed so the results are consistent
set.seed(1986, sample.kind = "Rounding")

# test set will be 20% of the edx set
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]

# Make sure userId, movieId and genre in test set are also in the train set
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId") %>%
  semi_join(train_set, by = "genres")
```

A loss function is defined to evaluate algorithm performance. The Residual Mean Squared Error is used to penalize larger errors.

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

Baseline

The simplest possible model is to predict the same rating for each movie. The estimate that minimizes the residual mean squared error with this approach is the average.

```r
mu <- mean(train_set$rating)
average_rmse <- RMSE(test_set$rating, mu)
```

The code below creates a table to store the RMSE results:

```r
rmse_results <- data_frame(method = "Average", RMSE = average_rmse)
rmse_results %>% knitr::kable()
```

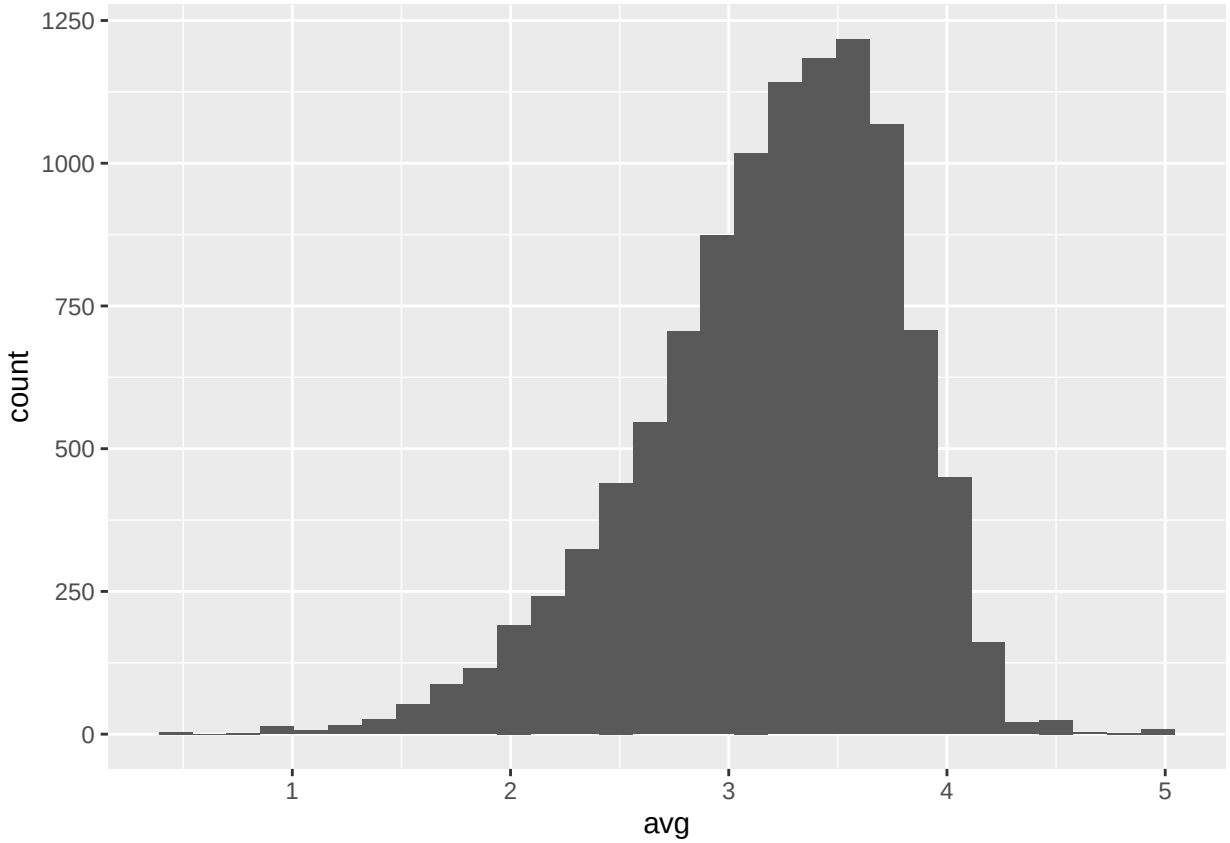
method	RMSE
Average	1.060226

Exploring Bias

Movie Bias

Different movies have very different average ratings. This makes sense - some movies are good, and some movies are bad. Good films will have higher average ratings and bad films will have low average ratings. The histogram below shows the skew of average ratings for the films in the dataset:

```r
movie_avg_ratings <- train_set %>% group_by(movieId) %>% summarize(avg = mean(rating))
movie_avg_ratings %>% ggplot(aes(avg)) + geom_histogram()
```



A "film bias" factor can be created for each film by taking the mean rating for each film and subtracting the overall mean rating across all ratings.

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

Accounting for film bias in the algorithm should help improve RMSE.

```
predicted_ratings_movie_effect <- mu + test_set %>% left_join(movie_bias, by = "movieId") %>% .$b_i
movie_bias_rmse <- RMSE(test_set$rating, predicted_ratings_movie_effect)
```

Here are the results compared:

```
rmse_results <- bind_rows(rmse_results, data_frame(method = "Movie Bias Model",
                                                   RMSE = movie_bias_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Average	1.0602257
Movie Bias Model	0.9437204

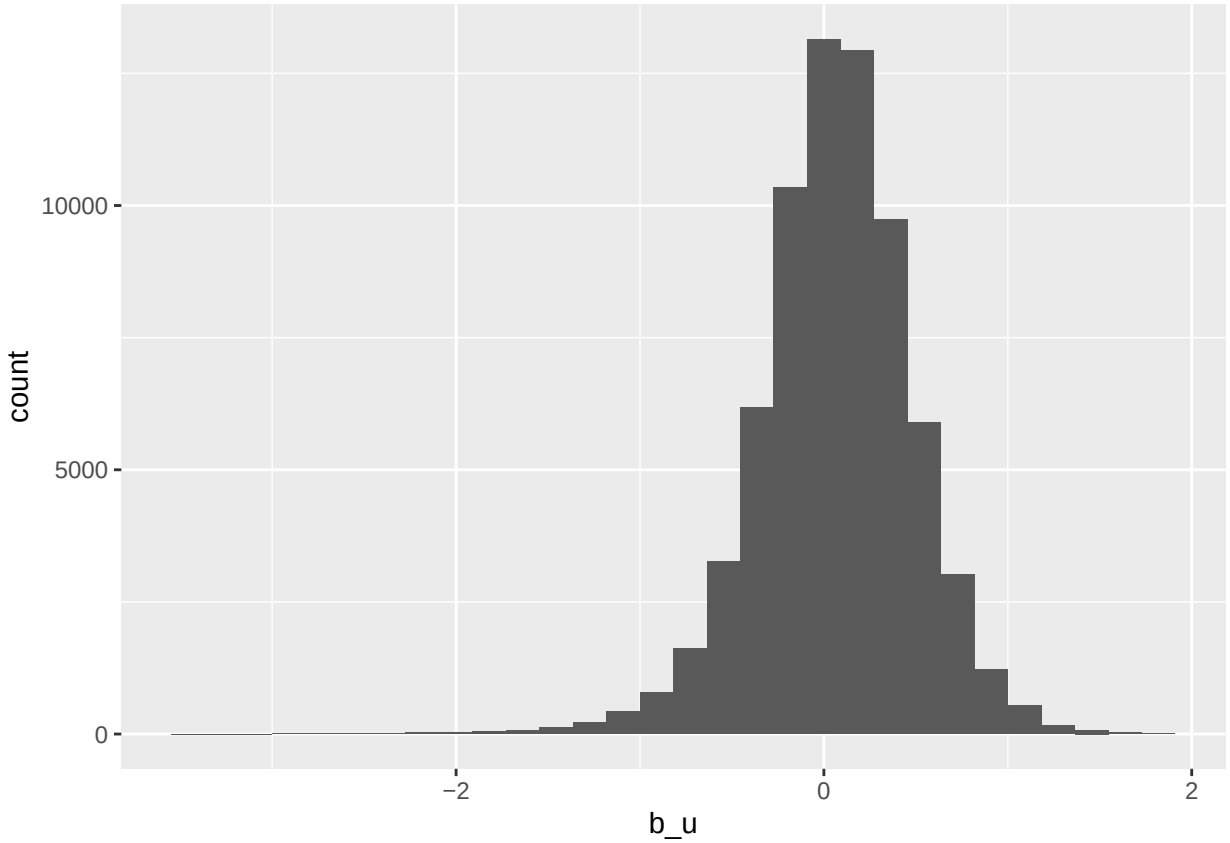
Adding the movie bias to the algorithm has improved overall accuracy.

User Bias

Different users have very different ratings (some consistently rate higher/lower than average). In evaluating user bias, movie bias is still incorporated because the goal is to assess how users rated each movie vs the average rating. Some users could have only rated bad movies, but that doesn't mean they are tough critics.

Below is a histogram of the user bias. The graph shows that some users do consistently rate movies higher than average (easy critics), and some users consistently rate movies lower than average (tough critics).

```
user_bias <- train_set %>% left_join(movie_bias, by='movieId') %>%
  group_by(userId) %>% summarize(b_u = mean(rating - mu - b_i))
user_bias %>% ggplot(aes(b_u)) + geom_histogram()
```



Accounting for user bias in the algorithm should help futher improve the RMSE of the predictions.

```
predicted_ratings_movie_and_user_effect <- test_set %>% left_join(movie_bias, by='movieId') %>%
  left_join(user_bias, by='userId') %>% mutate(prediction = mu + b_i + b_u) %>% .$prediction

movie_user_rmse <- RMSE(test_set$rating, predicted_ratings_movie_and_user_effect)
```

Here are the results compared:

```
rmse_results <- bind_rows(rmse_results, data_frame(method = "Movie + User Bias Model",
                                                   RMSE = movie_user_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Average	1.0602257
Movie Bias Model	0.9437204
Movie + User Bias Model	0.8663504

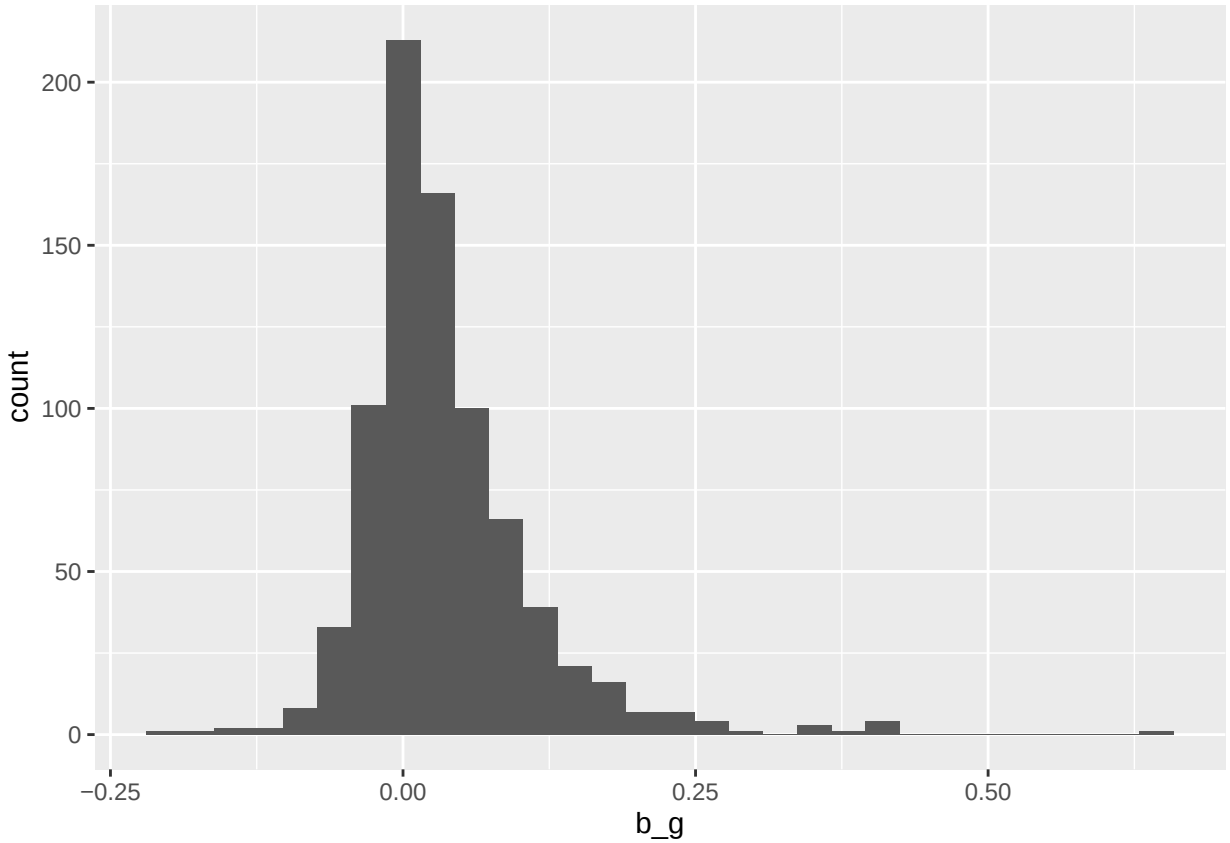
Adding the user bias to the model has further improved the predictions.

Genre Bias

Certain genres seem to have consistently lower ratings than others

Here is a histogram of the skew of average ratings for each genre:

```
genre_bias <- train_set %>% left_join(movie_bias, by='movieId') %>%
  left_join(user_bias, by="userId") %>% group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))
genre_bias %>% ggplot(aes(b_g)) + geom_histogram()
```



Accounting for genre bias in the algorithm could help futher improve the RMSE of the predictions.

```
predicted_ratings_genre_bias <- test_set %>% left_join(movie_bias, by='movieId') %>%
  left_join(user_bias, by='userId') %>% left_join(genre_bias, by='genres') %>% mutate(prediction = mu +

movie_genre_rmse <- RMSE(test_set$rating, predicted_ratings_genre_bias)
```

Here are the results compared:

```
rmse_results <- bind_rows(rmse_results, data_frame(method = "Movie + User + Genre Bias Model",  RMSE = r

rmse_results %>% knitr::kable()
```

method	RMSE
Average	1.0602257
Movie Bias Model	0.9437204
Movie + User Bias Model	0.8663504
Movie + User + Genre Bias Model	0.8660395

Adding the genre bias to the model has further improved the predictions.

There are potentially more features within the data that would contain a bias that could be used to improve predictions. Instead of exploring these further, this analysis will now focus on a technique called regularization to further improve the predictions.

Regularization

Regularization is useful for minimizing variability in small sample sizes. Within the data set, there are some films that have received many ratings, and some that have receieved very few. Likewise, some users have rated many titles, and some have rated only a few titles overall.

Motivation

Taking a closer look at the movie bias estimates, here is a list of the 10 titles with the highest (positive) movie bias:

```
# Add the movie titles so we can tell which movies have the highest and lowest bias
movie_titles <- edx %>%
  select(movieId, title) %>%
  distinct()

movie_bias %>% left_join(movie_titles, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, b_i) %>%
  slice(1:10) %>%
  knitr::kable()
```

title	b_i
Hellhounds on My Trail (1999)	1.48755
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.48755
Satan's Tango (Sátántangó) (1994)	1.48755
Shadows of Forgotten Ancestors (1964)	1.48755
Fighting Elegy (Kenka erejii) (1966)	1.48755
Sun Alley (Sonnenallee) (1999)	1.48755
Along Came Jones (1945)	1.48755

title	b_i
Angus, Thongs and Perfect Snogging (2008)	1.48755
Hunger (2008)	1.48755
Human Condition III, The (Ningen no joken III) (1961)	1.23755

This list doesn't match expectations. IMDb has a list of the top 250 films, but none of the top-rated films on IMDb appear in the top 10 movie bias list.

If the number of ratings for the titles with the highest movie bias is added, it can be seen that they each have very few ratings:

```
train_set %>% dplyr::count(movieId) %>%
  left_join(movie_bias) %>%
  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)	1.48755	1
Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	1.48755	2
Satan's Tango (Sátántangó) (1994)	1.48755	2
Shadows of Forgotten Ancestors (1964)	1.48755	1
Fighting Elegy (Kenka erejii) (1966)	1.48755	1
Sun Alley (Sonnenallee) (1999)	1.48755	1
Along Came Jones (1945)	1.48755	1
Angus, Thongs and Perfect Snogging (2008)	1.48755	1
Hunger (2008)	1.48755	1
Human Condition III, The (Ningen no joken III) (1961)	1.23755	2

Similarly, here is a list of the 10 titles with the lowest (negative) movie bias. These also have few ratings (with the exception of From Justin to Kelly, which is widely regared as one of the worst films ever made)

```
train_set %>% dplyr::count(movieId) %>%
  left_join(movie_bias) %>%
  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)	-3.012450	2
Grief (1993)	-3.012450	1
Confessions of a Superhero (2007)	-3.012450	1
War of the Worlds 2: The Next Wave (2008)	-3.012450	2
SuperBabies: Baby Geniuses 2 (2004)	-2.694269	44
Hip Hop Witch, Da (2000)	-2.666296	13
From Justin to Kelly (2003)	-2.646597	164

title	b_i	n
Disaster Movie (2008)	-2.554117	24
Criminals (1996)	-2.512450	1
Mountain Eagle, The (1926)	-2.512450	2

For the most part, the titles on these lists have been rated either very favorably or very negatively by a handful of users and it's not a good representation of the true quality of the film. If these titles had more ratings, it's unlikely the bias factors would continue to be so large.

To improve the predictions, a technique called regularization will be used. Regularization works by adding a constant (lambda) to the denominator when calculating averages. This has the effect of pulling the average towards zero when sample size is small, but has a nominal effect when sample size is large.

At first, to test the effect of regularization, the value for lambda will simply be selected. Later this parameter will be optimized to further improve results.

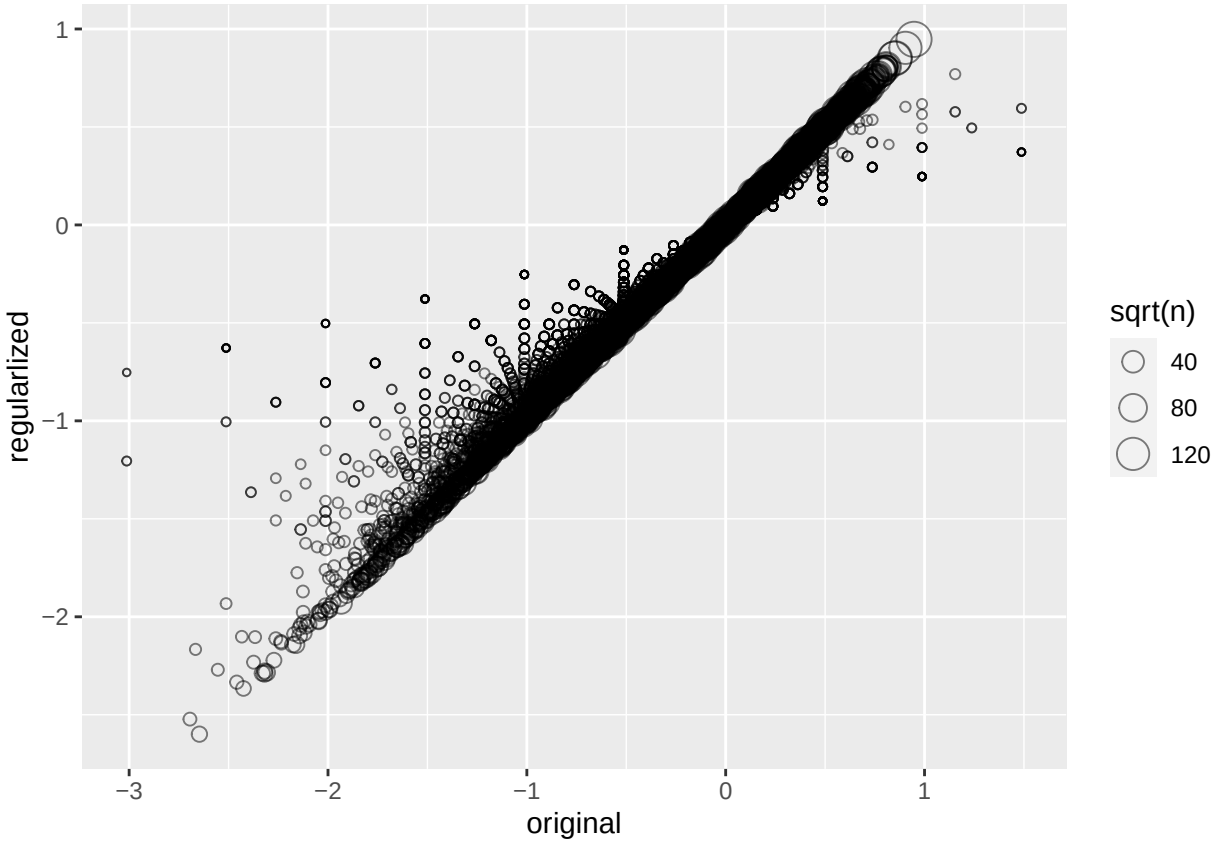
```
lambda <- 3
```

Adding lamdba to the denominator pulls the average rating towards zero for films with few ratings but has a nominal effect on films with many ratings. The code below shows how to calculate the regularized movie bias by adding the lambda parameter to the denominator.

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

The graph below visualizes the effect of regularization by plotting the original and regularized movie bias together. Movies that have fewer ratings have their bias normalized towards 0.

```
data_frame(original = movie_bias$b_i,
           regularlized = movie_reg_bias$b_i,
           n = movie_reg_bias$n_i) %>%
  ggplot(aes(original, regularlized, size=sqrt(n))) +
  geom_point(shape=1, alpha=0.5)
```



Here are the top 10 highest rated movies with regularize movie bias:

```
train_set %>%
  dplyr::count(movieId) %>%
  left_join(movie_reg_bias) %>%
  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)	0.9471175	22305
Godfather, The (1972)	0.9037857	14227
Usual Suspects, The (1995)	0.8526312	17385
Schindler's List (1993)	0.8487986	18620
Paths of Glory (1957)	0.8097389	1220
Double Indemnity (1944)	0.8078062	1702
Casablanca (1942)	0.8068783	9027
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	0.8063419	2367
Rear Window (1954)	0.8051705	6412
Seven Samurai (Shichinin no samurai) (1954)	0.8003583	4169

This list makes much more sense - it more closely matches IMDb's list of top films.

Similarly, here are the 10 lowest rated movies with regularized movie bias:

```
train_set %>%
  dplyr::count(movieId) %>%
  left_join(movie_reg_bias) %>%
  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)	-2.599053	164
SuperBabies: Baby Geniuses 2 (2004)	-2.522294	44
Pokémon Heroes (2003)	-2.365805	120
Carnosaur 3: Primal Species (1996)	-2.333851	56
Gigli (2003)	-2.290304	238
Pokemon 4 Ever (a.k.a. Pokémon 4: The Movie) (2002)	-2.287981	162
Glitter (2001)	-2.284097	284
Barney's Great Adventure (1998)	-2.277109	165
Disaster Movie (2008)	-2.270326	24
Faces of Death: Fact or Fiction? (1999)	-2.231703	47

This also makes sense - all of these titles appear in Wikipedia's list of worst films.

Using Regularized Movie Bias in predictions

Since the regularized movie bias gives a more accurate picture of the best/worst films, using this in our model should help us minimize RMSE.

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

movie_bias_regularized_rmse <- RMSE(test_set$rating, predicted_ratings_movie_bias_regularized)
```

Comparing the predictions without movie bias regularization to our predictions with movie bias regularization, it can be seen that regularization has decreased overall RMSE and improved the model.

```
movie_bias_comparison <- data_frame(method = "Movie Bias Model (non-regularized)", RMSE = movie_bias_rms
movie_bias_comparison <- bind_rows(movie_bias_comparison, data_frame(method = "Movie Bias Model (regula
movie_bias_comparison %>% knitr::kable()
```

method	RMSE
Movie Bias Model (non-regularized)	0.9437204
Movie Bias Model (regularized)	0.9436512

Optimizing regularization

In the regularization approach above, the lambda parameter was chosen arbitrarily. Optimizing this parameter will lead to even better results. To optimize this value, a range of values will be tested and the one that results in the lowest RMSE will be chosen

First a sequence of numbers to test is created, and a variable that has the sum of the residuals for each film.

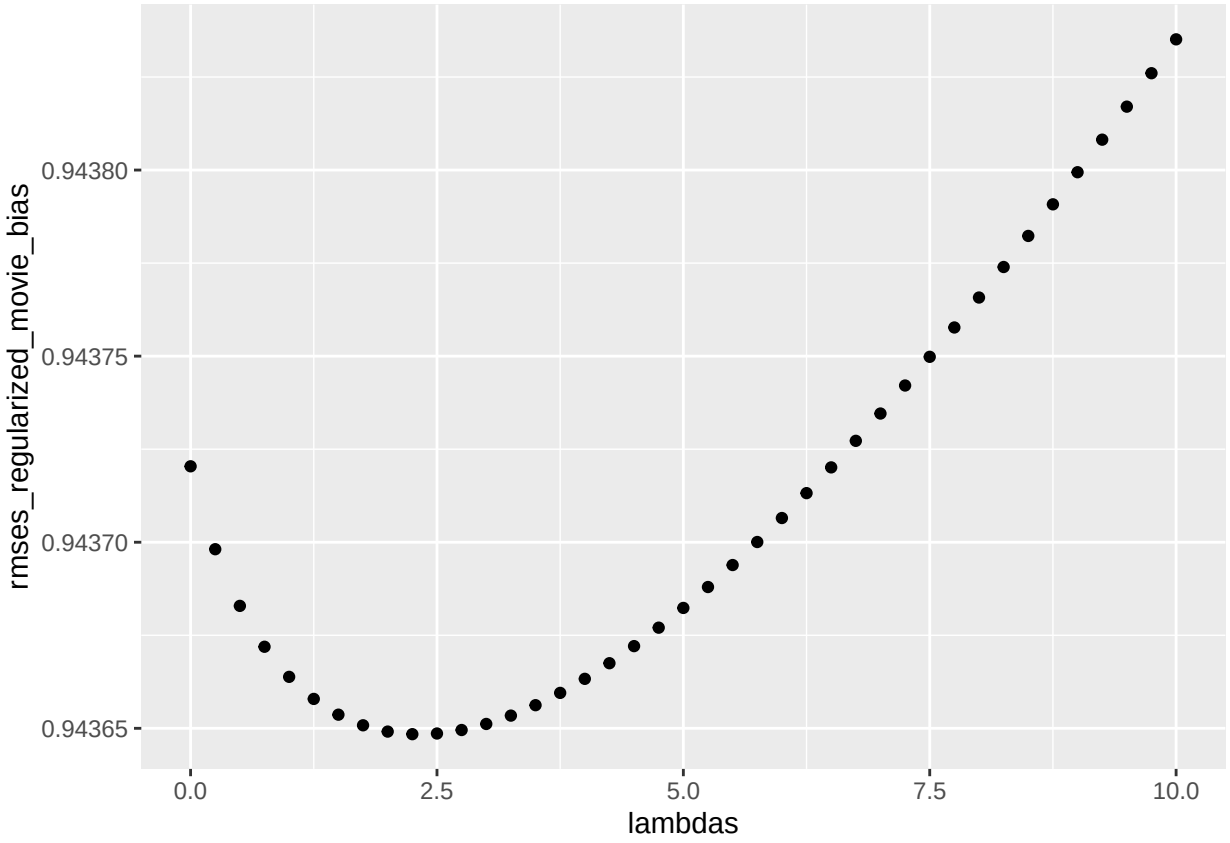
```
# Create a sequence of lambdas to test
lambdas <- seq(0, 10, 0.25)

sum_of_residuals <- train_set %>%
  group_by(movieId) %>%
  summarize(s = sum(rating - mu), n_i = n())
```

Next, a function is used to find the RMSE for all values

```
rmses_regularized_movie_bias <- sapply(lambdas, function(l){
  predicted_ratings <- test_set %>%
    left_join(sum_of_residuals, by='movieId') %>%
    mutate(b_i = s/(n_i+l)) %>%
    mutate(pred = mu + b_i) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})
```

The plot below shows the results with different values for lambda

```
qplot(lambdas, rmses_regularized_movie_bias)
```



The lambda value which minimized RMSE is 2.25, which results in an RMSE of 0.9436484 - lower than the previous value (when lamba was simply selected) and better than the non regularized movie bias model.

Best lambda:

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

```
## [1] 2.25
```

RMSE:

```
movie_bias_regularized_optimized_rmse <- rmses_regularized_movie_bias[which.min(rmses_regularized_movie
movie_bias_regularized_optimized_rmse
```

```
## [1] 0.9436484
```

RMSE results table:

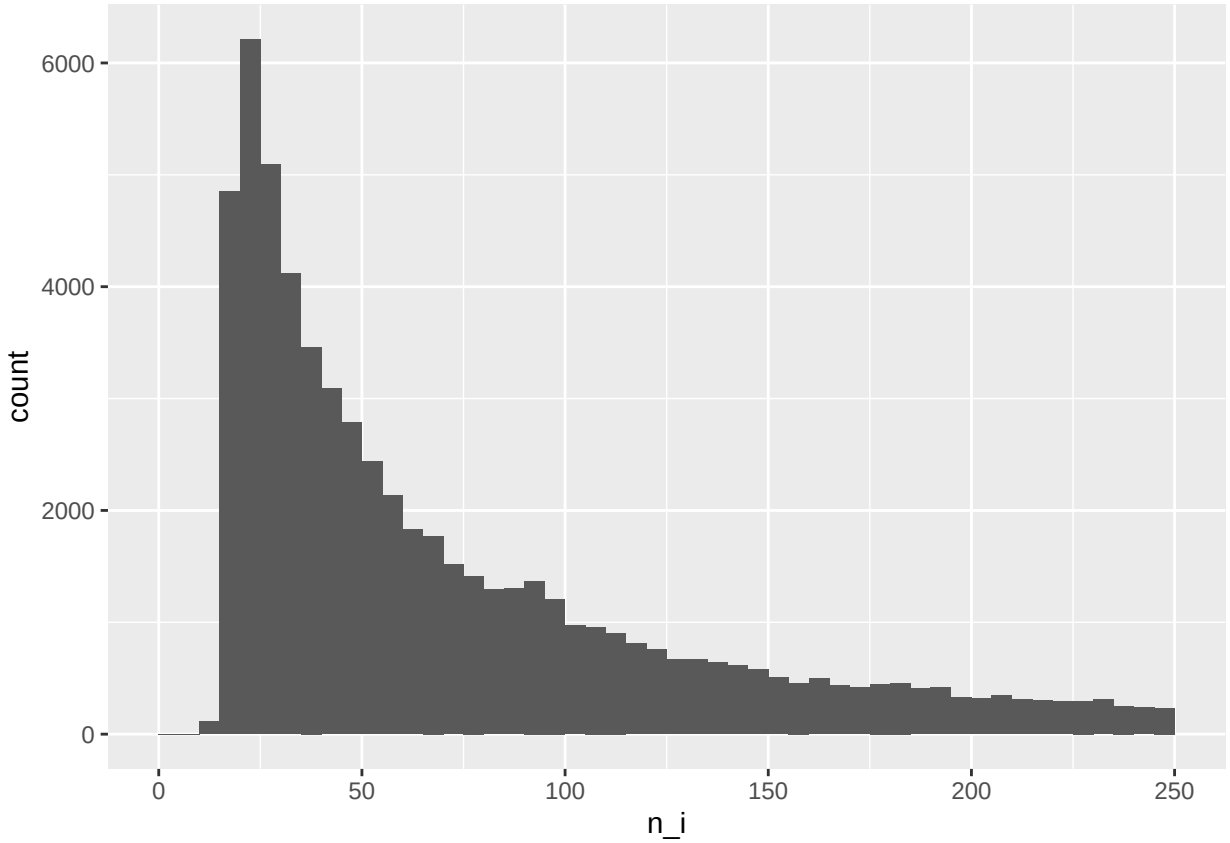
```
movie_bias_comparison <- bind_rows(movie_bias_comparison, data_frame(method = "Movie Bias Model (regula
movie_bias_comparison %>% knitr::kable()
```

method	RMSE
Movie Bias Model (non-regularized)	0.9437204
Movie Bias Model (regularized)	0.9436512
Movie Bias Model (regularized + optimized)	0.9436484

Regularizing User Bias

Some users have rated many films, but some have rated only a few films:

```
edx %>% group_by(userId) %>% summarize(n_i = n()) %>% ggplot(aes(n_i)) + geom_histogram(breaks=c(seq(0,
```



Some of the users with the highest user bias have relatively few ratings

```
train_set %>% dplyr::count(userId) %>%
  left_join(user_bias) %>%
  arrange(desc(b_u)) %>%
  select(userId, b_u, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "userId"
```

userId	b_u	n
13524	1.880799	18
56965	1.821122	25
7999	1.808953	28
45895	1.792459	12
18591	1.779548	15
36022	1.748085	68
52749	1.738764	66
46484	1.722574	18
55534	1.711688	34
36896	1.709252	117

The same is true for some of the users with the lowest user bias

```
train_set %>% dplyr::count(userId) %>%
  left_join(user_bias) %>%
  arrange(b_u) %>%
  select(userId, b_u, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "userId"
```

userId	b_u	n
13496	-3.394712	16
48146	-3.256304	25
49862	-3.099990	13
63381	-2.966776	14
6322	-2.957194	13
62815	-2.919022	15
15515	-2.694347	22
6907	-2.688090	19
43628	-2.638439	15
42019	-2.561650	20

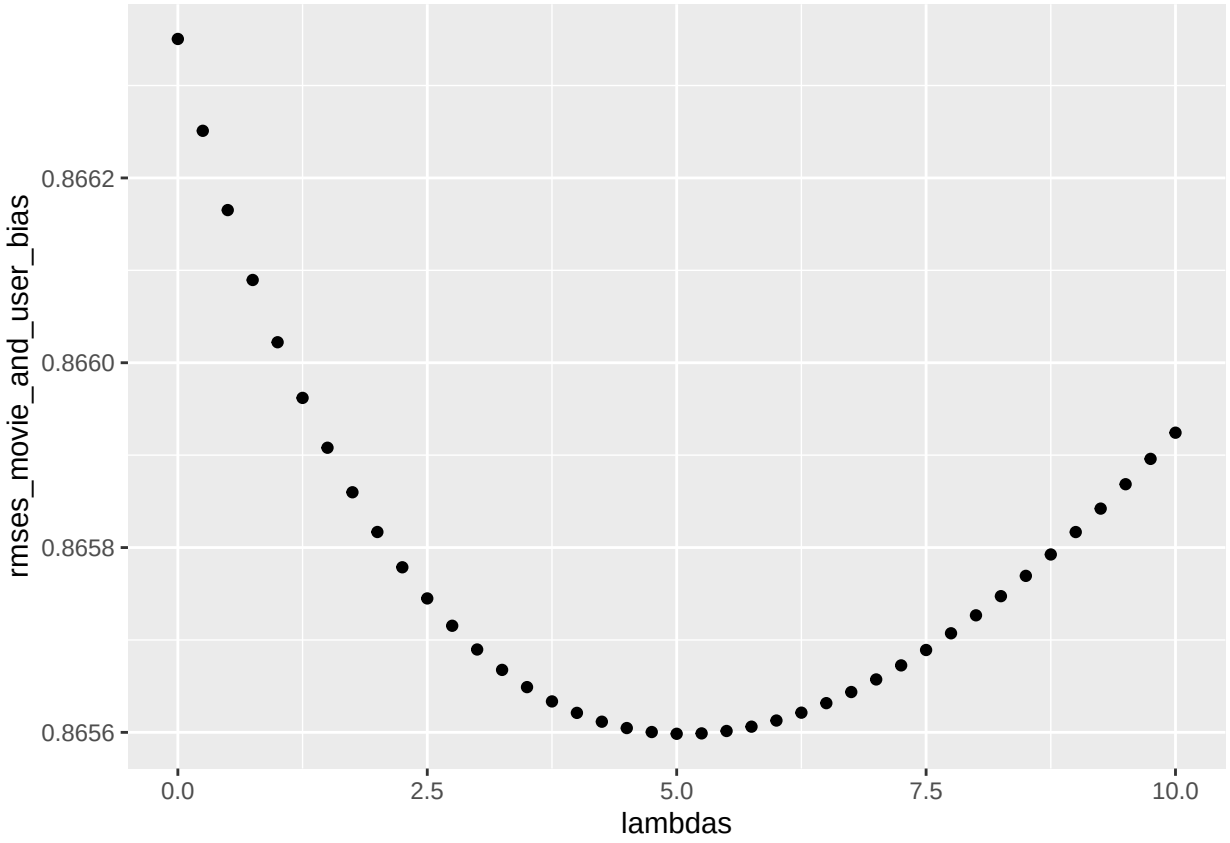
This makes user bias another good candidate for regularization. Regularizing the user bias can help further minimize RMSE.

The function below gets the RMSE for the predictions with regularized movie bias and regularized user bias across the range of lambdas defined earlier.

```
rmses_movie_and_user_bias <- sapply(lambdas, function(l){
  # regularized movie effect
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  # regularized user effect
  b_u <- train_set %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  # predicted rating
  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))
})
```

Plotting the results shows the RMSE values for each lambda

```
qplot(lambdas, rmses_movie_and_user_bias)
```



The lambda value which minimized RMSE is 5, which results in an RMSE of 0.8655985, which is lower than the movie + user bias model without regularization.

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

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

```
movie_and_user_bias_regularized_rmse <- rmses_movie_and_user_bias[which.min(rmses_movie_and_user_bias)]

movie_and_user_bias_comparison <- data_frame(method = "Movie + User Bias Model", RMSE = movie_user_rmse)
movie_and_user_bias_comparison <- bind_rows(movie_and_user_bias_comparison, data_frame(method = "Movie +
movie_and_user_bias_comparison %>% knitr::kable()
```

method	RMSE
Movie + User Bias Model	0.8663504
Movie + User Bias Model (regularized + optimized)	0.8655985

Regularizing Genre Bias

Similar to Movie and User Bias, many of the Genres with the highest and lowest bias have relatively few

titles that fall within the genre

Highest bias:

```
train_set %>% dplyr::count(genres) %>%
  left_join(genre_bias) %>%
  arrange(desc(b_g)) %>%
  select(genres, b_g, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "genres"
```

genres	b_g	n
Fantasy|Mystery|Sci-Fi|War	0.6354588	2
Adventure|Fantasy|Film-Noir|Mystery|Sci-Fi	0.4227809	2
Comedy|Fantasy|Mystery|Sci-Fi	0.4142763	4
Action|Fantasy|Romance	0.4119062	8
Action|Animation|Comedy|Horror	0.4038233	2
Action|Crime|Film-Noir	0.3680653	11
Adventure|Comedy|Drama|Fantasy|Mystery|Sci-Fi	0.3607499	6
Action|War|Western	0.3552910	1
Drama|Musical|Thriller	0.3440924	2
Action|Comedy|Romance|Western	0.2922779	9

Lowest bias:

```
train_set %>% dplyr::count(genres) %>%
  left_join(genre_bias) %>%
  arrange(b_g) %>%
  select(genres, b_g, n) %>%
  slice(1:10) %>%
  knitr::kable()
```

```
## Joining, by = "genres"
```

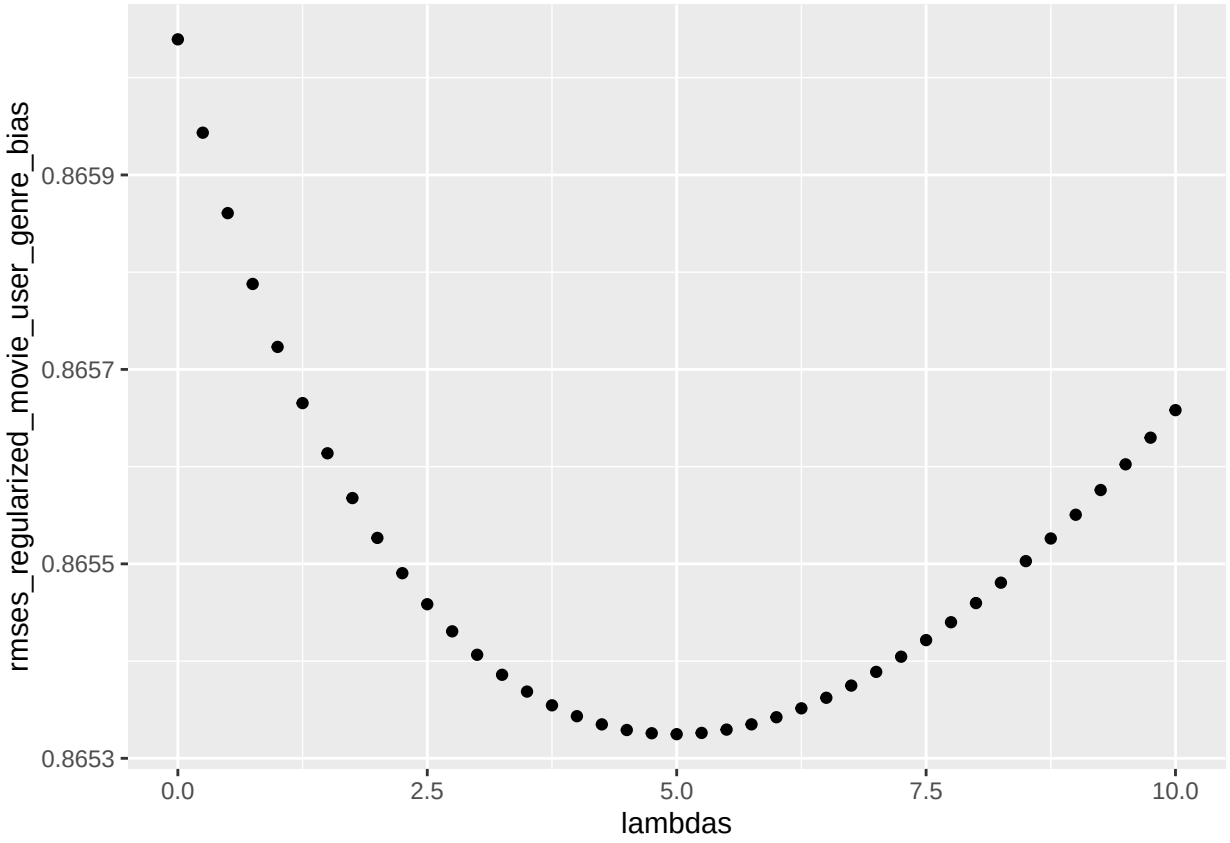
genres	b_g	n
Action|Drama|Musical|Romance	-0.2137206	12
Action|Adventure|Fantasy|War	-0.1623932	39
Adventure|Children|Comedy|Fantasy|Romance	-0.1351544	758
Adventure|Comedy|Drama|Fantasy	-0.1320720	23
Adventure|Horror|Romance|Sci-Fi	-0.1264047	3
Animation|Children|Comedy|Romance	-0.1036559	1233
Action|Adventure|Crime|Mystery|Thriller	-0.0989257	6
Adventure|Children|Romance	-0.0852817	2906
Action|Adventure|Animation|Children|Fantasy	-0.0843419	576
Comedy|Crime|Fantasy	-0.0814301	47

Regularizing the genre bias will help further improve RMSE.

```
# This portion may take a few minutes to run

rmses_regularized_movie_user_genre_bias <- sapply(lambdas, function(l){
  # regularized movie effect
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  # regularized user effect
  b_u <- train_set %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  #regularized genre effect
  b_g <- train_set %>%
    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 rating
  predicted_ratings <-
    test_set %>%
    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) %>%
    .$pred
  return(RMSE(predicted_ratings, test_set$rating))
})
```

Plotting the RMSEs for each lambda shows which value minimized RMSE:

```
qplot(lambdas, rmses_regularized_movie_user_genre_bias)
```



The best lambda value is:

```
best_lambda <- lambdas[which.min(rmses_regularized_movie_user_genre_bias)]
best_lambda
```

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

And the final RMSE is:

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

method	RMSE
Average	1.0602257
Movie Bias Model	0.9437204
Movie + User Bias Model	0.8663504
Movie + User + Genre Bias Model	0.8660395
Regularized Movie + User + Genres Bias Model	0.8653248

The Regularized Movie/User/Genres Bias model is the most accurate model so far.

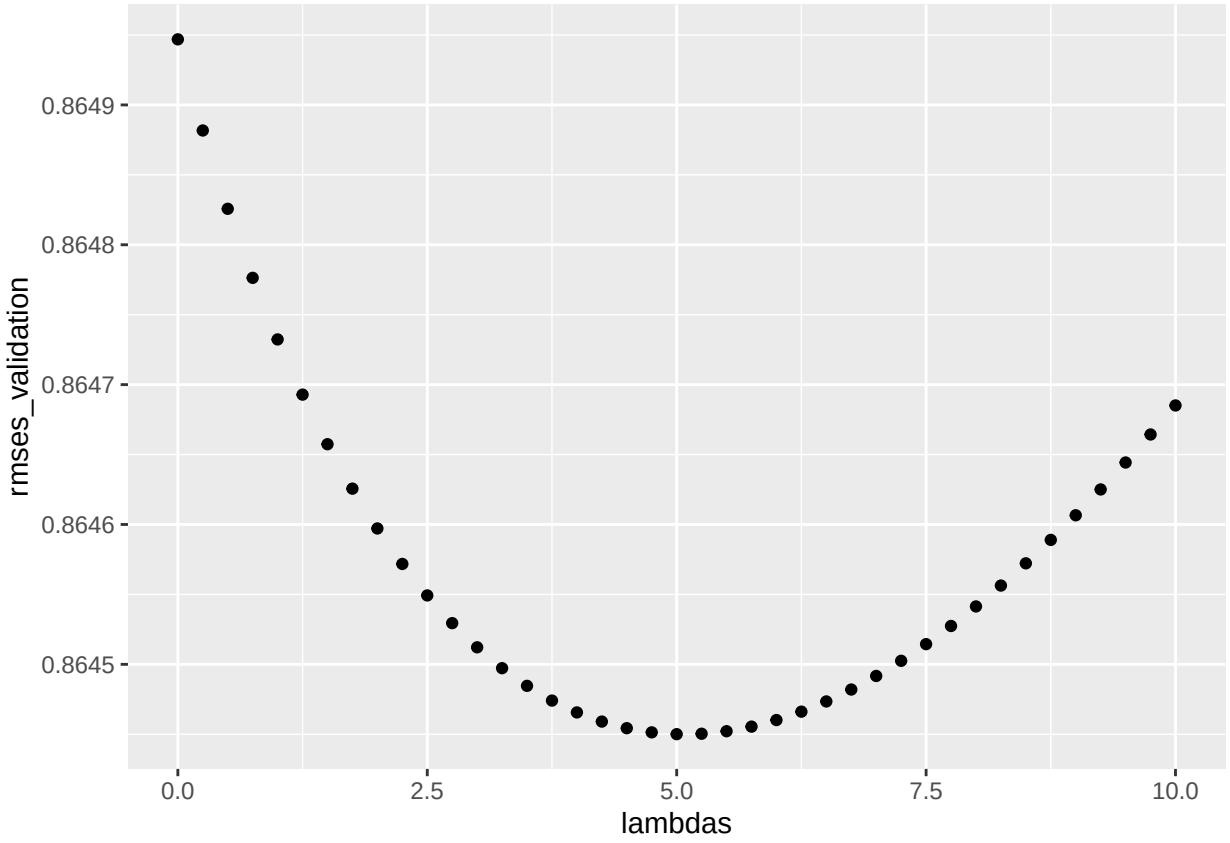
Results

Now the most accurate model will be tested on the validation set to judge evaluate overall accuracy. In this case, the full edx dataset is used to train the model.

```
# This portion may take a few minutes to run

rmses_validation <- sapply(lambdas, function(l){
  # average rating
  mu <- mean(edx$rating)
  # regularized movie effect
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))
  # regularized user effect
  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
  #regularized genre effect
  b_g <- edx %>%
    left_join(b_i, by="movieId") %>%
    left_join(b_u, by="userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - b_i - b_u - mu)/(n()+l))
  # predicted rating
  predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    mutate(pred = mu + b_i + b_u + b_g) %>%
    .$pred
  return(RMSE(predicted_ratings, validation$rating))
})
```

Plotting the RMSEs shows the best lambda value:

```
qplot(lambdas, rmses_validation)
```



The final RMSE is:

```
min(rmses_validation)
```

```
## [1] 0.8644501
```

Conclusion

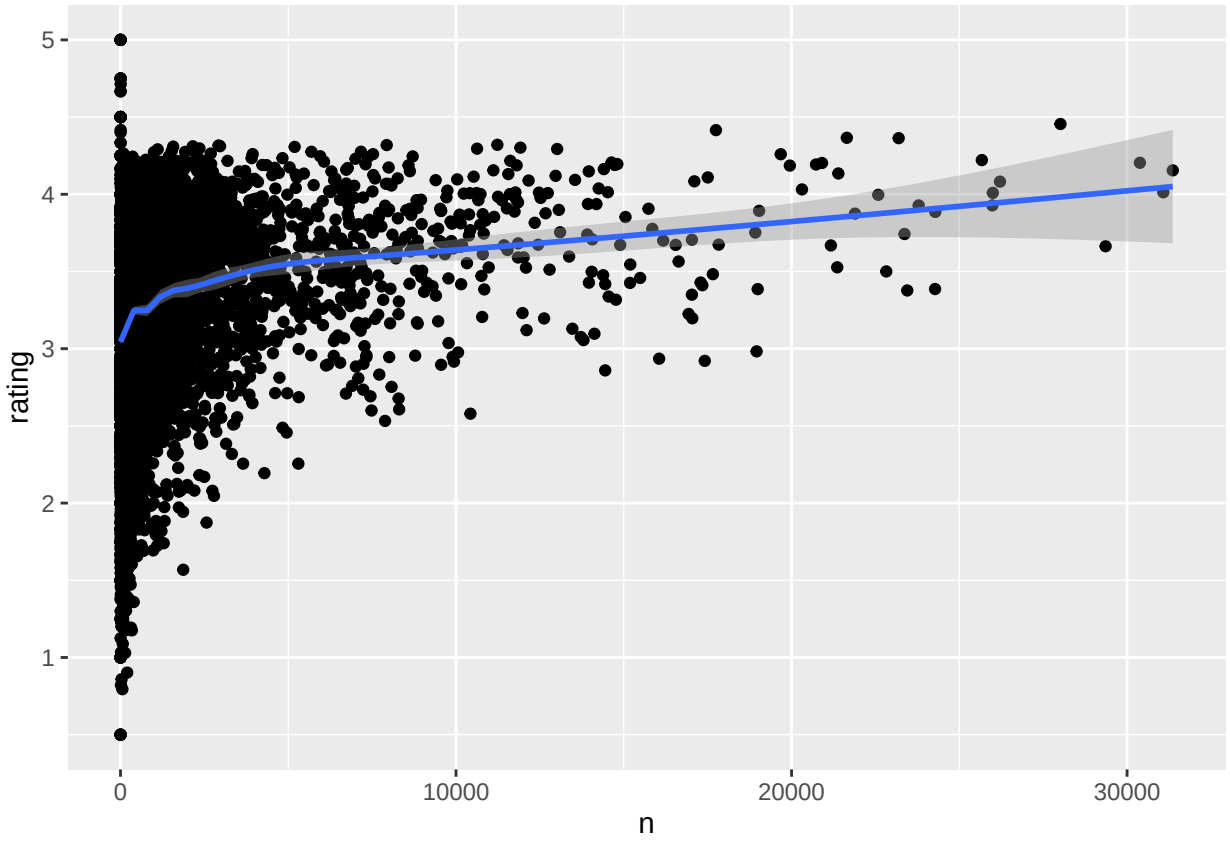
Summary

By incorporating bias into the prediction algorithm, and using regularization to reduce variability for small sample sizes, the RMSE of the predicted ratings was significantly reduced. The final model performs much better than just predicting the overall average (guessing).

Limitations & Future Work

While this analysis considered three biases within the data, there are other biases that could potentially be incorporated into the prediction algorithm to further reduce RMSE. One example is the total number of ratings.

```r
edx %>%
    group_by(movieId) %>%
    summarize(n = n(),
              title = title[1],
              rating = mean(rating)) %>%
    ggplot(aes(n, rating)) +
    geom_point() +
    geom_smooth()
```



The plot above shows that movies with more ratings have a higher average rating overall. One possible explanation for this could be that people prefer to watch well-rated movies, which ultimately then get more ratings.

There are other tactics that could be employed to improve predictions - including matrix factorization and principal components analysis.

Matrix factorization and principle component can help identify structure in the data based on the relationship between films including patterns like "people that like ganster movies, will like other gangster movies and people who don't like a ganster movie are less likely to like other ganster movies". However, this type of analysis is complicated to model and requires intensive computing resources.