# MovieLens Project Report

Bill James / jamesbcdl@gmail.com (mailto:jamesbcdl@gmail.com)

April 14, 2019

**Executive Summary, Introduction and Overview**

This document describes the training of a machine learning algorithm on a portion of the MovieLens dataset so that it will accurately predict movie ratings on a second or test subset of that data. The goal of the analysis is to establish an algorithm with a residual means squared error (RMSE) of 0.87750 or lower.

The approach used in this analysis was to:

- Download the 10M version of the MovieLens dataset (movies, ratings, and tags)
- Combine the ratings and movies sets into a single dataset for analysis
- Split out a training subset (9,000,055 ratings) and a test subset (999,999 ratings)
- Successively improve an algorithm so it produced an RMSE below 0.87750
- Verify that the algorithm produced similar (or better) results on the test dataset

In training the algorithm, the following steps were taken:

- Establishing a baseline RMSE via the simple average rating for all movies
- Introducing a beta (b_i), to account for general bias in movie ratings
- Introducing a beta (b_u), to account for user-specific rating effects
- Testing to see if regularization would improve the model results
- Using cross-validation to determine if that could improve the results

The process above did produce a final RMSE of 0.8566952 for the training set and 0.8251770 for the test set. Eliminating movie rating bias seemed to have the most impact on the training set. Movie bias and user bias seemed to have about equal impact on the test set. Regularization did not improve the results, due, it is assumed, to the fact that no movie had fewer than 20 users rating it, thereby mitigating the effect of very few users skewing the ratings with very high or very low ratings. Cross-validation produced a small improvement in the training data set, but not in the test set.

**The Dataset**

The source dataset contained 10,000,054 user ratings applied to 10,681 movies by 10,677 users. Users were selected at random and had rated at least 20 movies. Each movie and user was represented by a unique Id.

The source data were contained in three files: movies.dat, ratings.dat and tags.dat. For this analysis, the userId, movieId, rating, and timestamp fields were selected from the ratings data, and the movieId, title, and genres fields from the movies data. These were then joined on movieId to create the dataset for the analysis.

**The Approach**

The analysis used the residual mean squared error (RMSE) as a measure of how well the algorithm could predict a movie rating. RMSE represents the degree of error made when making such a prediction. The RMSE formula takes the square root of the average of the true ratings minus the predicted ratings squared, or:

sqrt(mean((true_ratings - predicted_ratings)^2))

To establish a baseline, the analysis calculated the *average* rating for all movies (3.512465) and its corresponding RMSE (1.060331). The challenge then was to see if it were possible to lower the RMSE value.

The analysis followed these steps to improve on the baseline:

1. *Account for the general bias in rating movies* - Different movies are rated differently. For example, blockbusters might have been rated higher because of the bandwagon effect. The analysis introduced The term, b_i to represent the average rating for a given movie i. This was used to mitigate that rating bias.
2. *Account for the general biases of users rating movies* - Different users have different tastes and biases. The analysis introduced the term b_u to account for the tendancy of a given user to rate a certain genre (or specific movie) significantly higher or lower than others might have.
3. *Determine if some ratings are unusually high or low* - Movies with fewer ratings might have unusually high or low ratings; this would affect the RMSE. The analysis used *regularization* to test for this. In theory, this would not have been a factor because no movie had fewer than 20 ratings.
4. *Use cross-validation to see if there is some value of lambda that will improve the RMSE* - The analysis tested to see if cross-validation could suggest a lambda value that would lower the RMSE to any significant degree.

**Training the Algorithm on the Training (edx) Dataset**

*Step 1: Introduce b_i to account for general bias in movie ratings*

In the code below, the term b_i was established for each movie by taking the average of each rating for that movie minus the average rating for all movies. This generated a revised rating prediction (and RMSE) with this bias accounted for:

```
# Calculate b_i as the average of a given movie's ratings minus the overall average
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

# Generate predicted ratings with b_i factored in
predicted_ratings <- mu + edx %>%
  left_join(movie_avgs, by = 'movieId') %>%
  pull(b_i)

# Calculate the resulting RMSE
model_1_rmse <- RMSE(predicted_ratings, edx$rating)
```

This was effective; the RMSE dropped considerably:

| method | RMSE |
|---|---:|
| Just the average | 1.0603313 |
| Movie Effect Model | 0.9423475 |

*Step 2: Introduce b_u to account for user bias in ratings*

The second step was to mitigate the effect of user biases in making ratings. As in the case above, the analysis introduced a term (in this case, b_u) to account for this. The term was calculated by taking the average of a given movie’s ratings minus the overall ratings average minus the movie bias influence (b_i).

```r
# Calculate b_u as the average of a given movie's ratings minus the overall average
# minus the general ratings bias
user_avgs <- edx %>%
  left_join(movie_avgs, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

# Generate predicted ratings with b_u factored in
predicted_ratings <- edx %>%
  left_join(movie_avgs, by = 'movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

# Calculate the new RMSE
model_2_rmse <- RMSE(predicted_ratings, edx$rating)
```

The resulting RMSE dropped further, although to a lesser degree.

| method | RMSE |
|---|---:|
| Just the average | 1.0603313 |
| Movie Effect Model | 0.9423475 |
| Movie + User Effects Model | 0.8567039 |

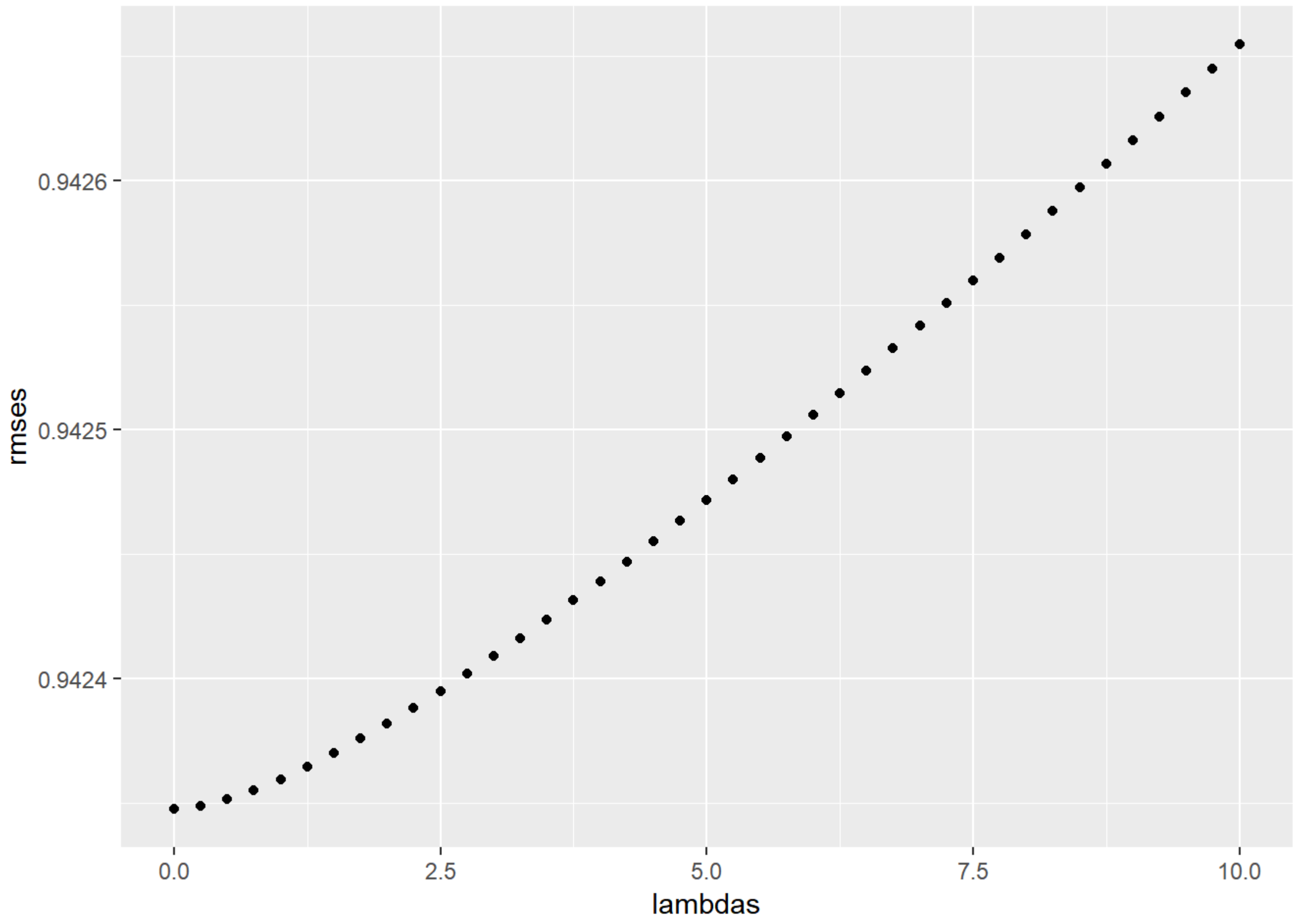
Taken together, these two adjustments met the goal of an RMSE of 0.87550 or lower. However, before moving on to test the algorithm against the test set, the analysis utilized regularization and cross-validation to determine if the RMSE could be lowered further.

*Step 3: Regularization*

Larger estimates of b_i, negative or positive, are more likely when fewer users rate movies. Large errors can increase the residual mean squared error. Regularization penalizes large estimates that come from small sample sizes by adding a penalty to the equations an algorithm is trying to minimize.

The anlysis looked for a penalty value (lambda) using this code:

```r
# Select a Lambda
lambdas <- seq(0, 10, 0.25)

# Again, the average rating for all movies
mu <- mean(edx$rating)

# Take the sum of each of the movie ratings minus the average rating
# and calculate the number of ratings for that movie
just_the_sum <- edx %>%
  group_by(movieId) %>%
  summarize(s = sum(rating - mu), n_i = n())

# Join this tibble to the training set, calc new b_i and prediction
rmses <- sapply(lambdas, function(l){
  predicted_ratings <- edx %>%
    left_join(just_the_sum, by = 'movieId') %>%
    mutate(b_i = s/(n_i+l)) %>%
    mutate(pred = mu + b_i) %>%
    pull(pred)
  return(RMSE(predicted_ratings, edx$rating))
})
```



The result was that the optimal value of lambda was 0 - implying that regularization would not affect our results, and that there are no unusually large or small ratings coming from small sample sizes. Running the RMSE again for completeness, the anlysis confirmed that the two steps already taken have resulted in the lowest RMSE.

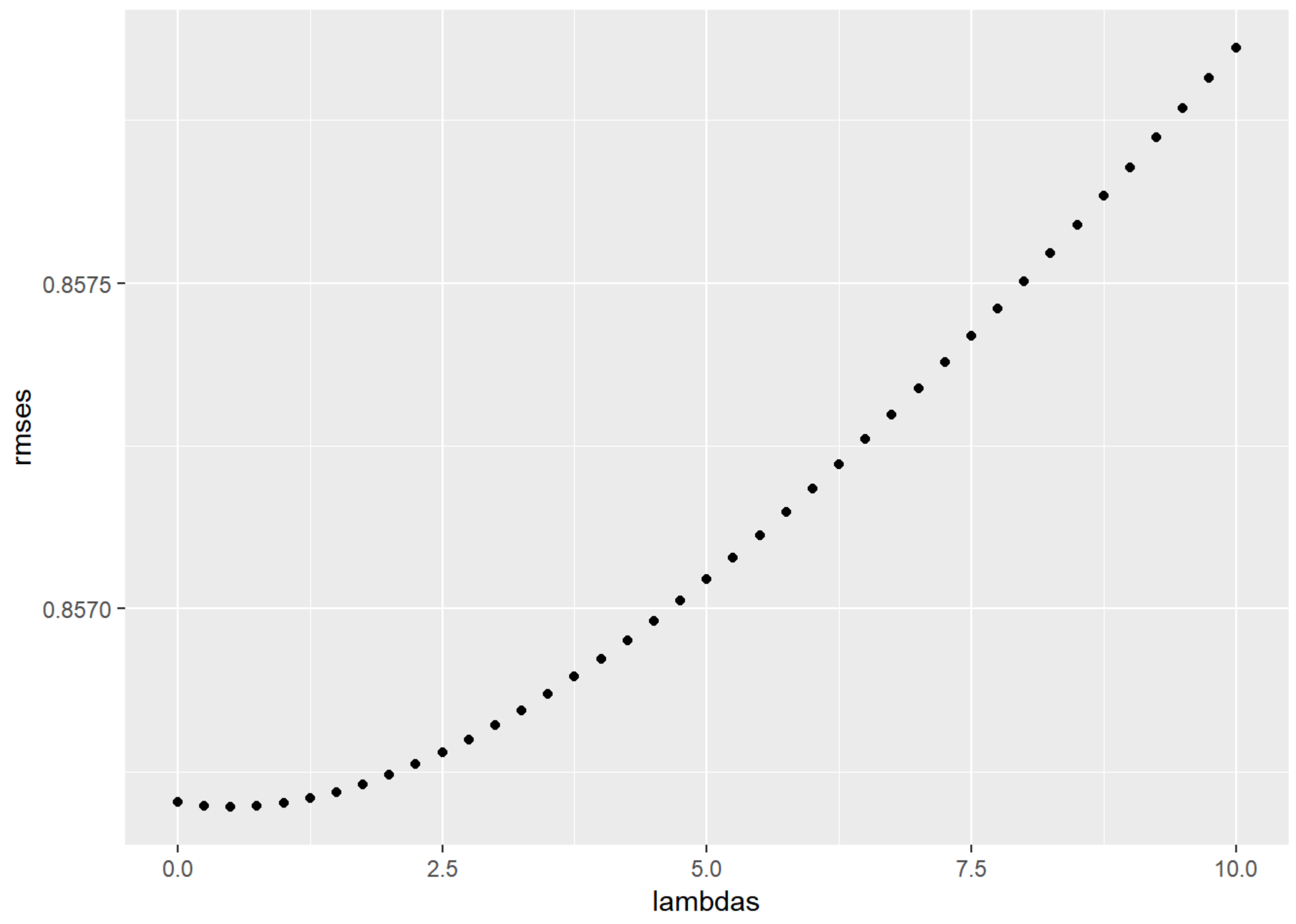
| method | RMSE |
|:---|---:|
| Just the average | 1.0603313 |
| Movie Effect Model | 0.9423475 |
| Movie + User Effects Model | 0.8567039 |
| Regularized Movie Effect Model | 0.9423475 |

*Step 4: Using Cross-valication to See if the RMSE Can be Improved*

For the sake of thoroughness, the analysis took one additional step, using cross-validation to determine if there was a lambda that would improve the results using this code:

```
# As before, select a Lambda
lambdas <- seq(0, 10, 0.25)

# Function to join the b_i and b_u values and calculate the new predicted ratings
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 <-
    edx %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)

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

The results suggest that a lambda of 0.5 might improve our results - the lowest RMSE calculated from the range of lambdas is 0.8566952, a tiny fraction better than the *Movie + User Effects* model.



| method | RMSE |
|---|---|
| Just the average | 1.0603313 |
| Movie Effect Model | 0.9423475 |
| Movie + User Effects Model | 0.8567039 |
| Regularized Movie Effect Model | 0.9423475 |
| Using cross-validation | 0.8566952 |

**Testing the Algorithm on the Test (validation) Dataset**

The anlysis then repeated the steps above to the test dataset, to ensure that the results did not differ significantly from the training dataset (for example, that there was no overfitting of the model).

*Step 1* - introducing the general movie ratings bias term b_i produced a result identical to the training set data:

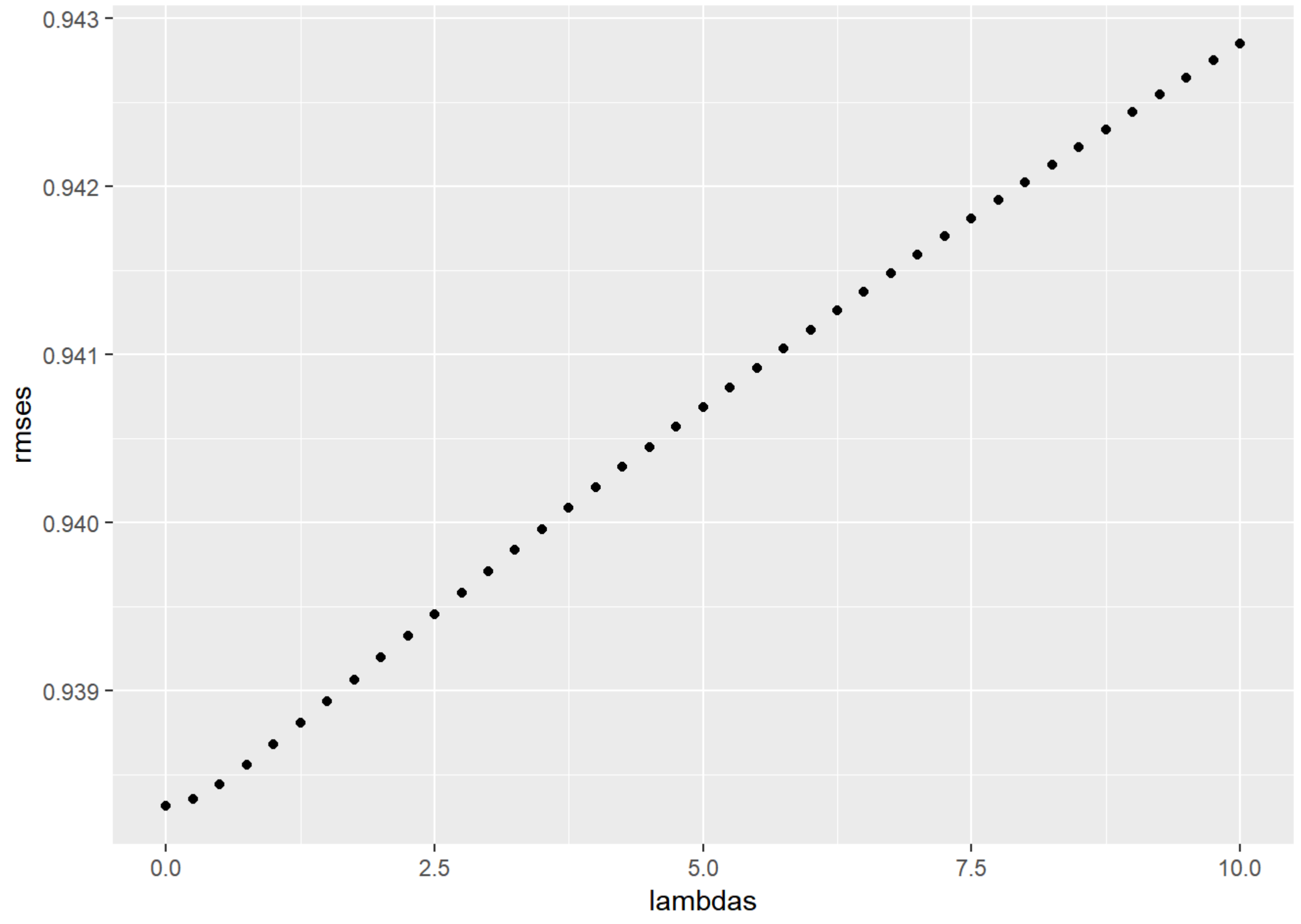
| method | RMSE |
|---|---|
| Just the average (Test Data) | 1.0612017 |
| Movie Effect Model (Test Data) | 0.9423475 |

*Step 2* - introducing the user bias term, b_u, produced a (very) small improvement over the training dataset:

| method | RMSE |
|---|---|

| method | RMSE |
|---|---|
| Just the average (Test Data) | 1.0612017 |
| Movie Effect Model (Test Data) | 0.9423475 |
| Movie + User Effects Model (Test Data) | 0.8251770 |

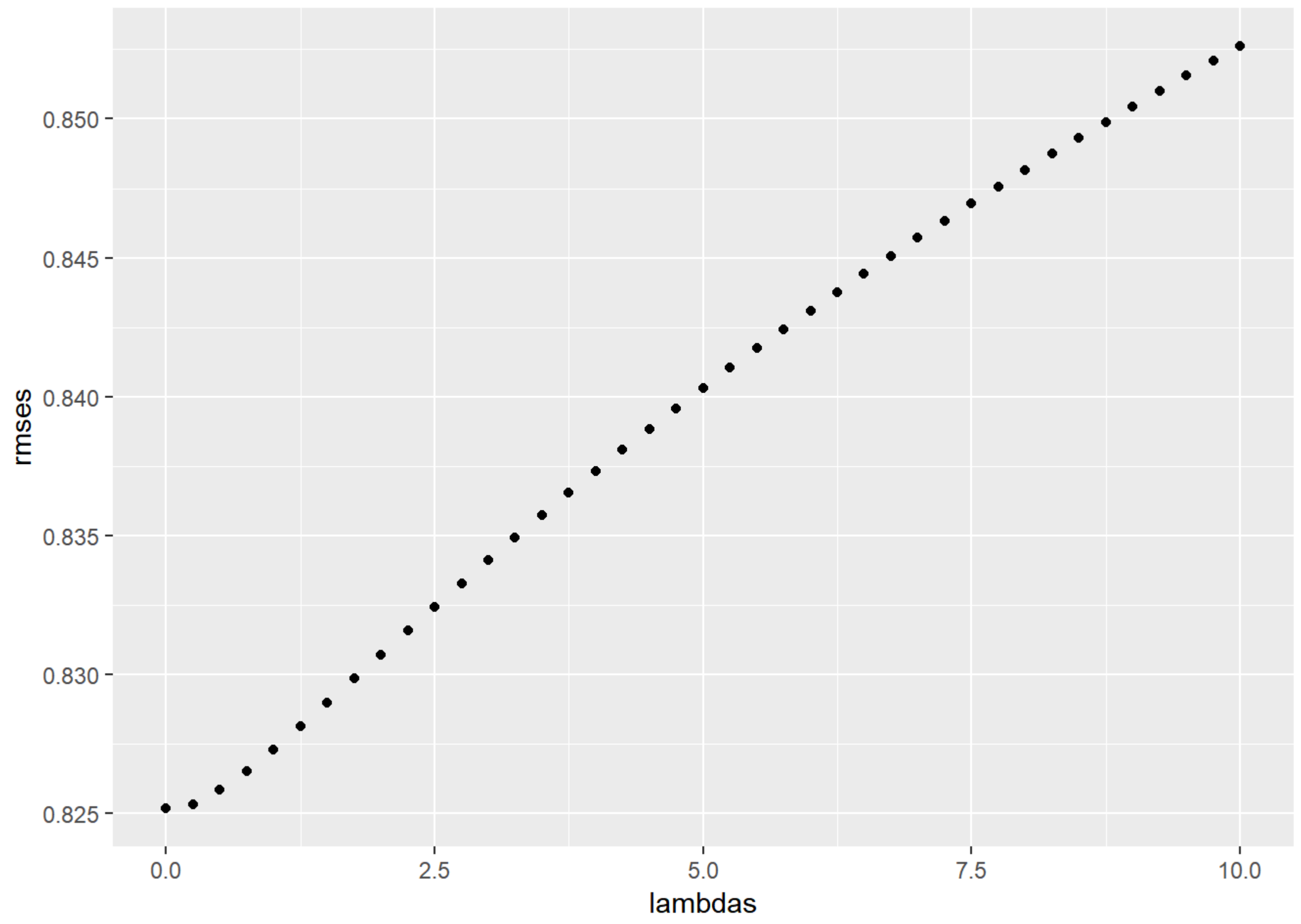
*Step 3* - as before, the analysis sought to see if regularization would improve the results. However, given the outputs from the training data, the assumption was that it would not, and this turned out to be the case. The resulting RMSE was 0.938 - not quite as good as the user effects model.



```
## [1] "The optimal value of lambda is 0"
```

| method | RMSE |
|---|---|
| Just the average (Test Data) | 1.0612017 |
| Movie Effect Model (Test Data) | 0.9423475 |
| Movie + User Effects Model (Test Data) | 0.8251770 |
| Regularized Movie Effect Model (Test Data) | 0.9383091 |

*Step 4* - finally, the analysis checked to see if cross-validation would discover a lambda that would improve the RMSE. The optimal lambda turned out to be zero, and returned results equal to the *Movie + User Effects Model*.



**Results**

The final results are shown in the training set and test set tables of RMSE results below.

- The movie effect model produced the most significant improvements in RMSE in both training and test sets
- The user effects model produced improvements in both datasets, but it was more pronounced in the test set
- Regularization did not appear to improve the results in either dataset
- Cross-validation produced a *very* slight improvement in the training set, and had no effect on the test set

Training Set

| method | RMSE |
|---|---:|
| Just the average | 1.0603313 |
| Movie Effect Model | 0.9423475 |
| Movie + User Effects Model | 0.8567039 |
| Regularized Movie Effect Model | 0.9423475 |
| Using cross-validation | 0.8566952 |

Test Set

| method | RMSE |
|---|---:|
| Just the average (Test Data) | 1.0612017 |

| method | RMSE |
|---|---|
| Movie Effect Model (Test Data) | 0.9423475 |
| Movie + User Effects Model (Test Data) | 0.8251770 |
| Regularized Movie Effect Model (Test Data) | 0.9383091 |
| Using Cross-validation (Test Data) | 0.8251770 |

**Conclusions**

- The movielens data is relatively high quality. Because no movie has fewer than 20 ratings, there appears to be little or no skew from a few unusually high or low ratings.
- Bias does exist in both the movie ratings themselves and in the users rating these movies. The RMSE dropped significantly when these biases were accounted for.
- An interesting follow-on analysis would be to determine if these biases vary by genre, or if they are general characteristics of all movie ratings.

[End]