HarvardX Professional Certificate in Data Science (PH125.9x) - MovieLens Capstone Assignment

Rob Costello

23/03/2021

Contents

1 Overview

This report artifact details the analysis of the MovieLens 10M dataset as part of the first capstone assignment in the HarvardX Professional Certificate in Data Science (PH125.9x) course. In this report I will summarise the goal of the project and the approach taken to perform the dataset analysis, followed by the final results of the analysis and a conclusion.

1.1 Introduction

A Recommendation System allows us to generate (predict) a recommendation for an entity (such as a user) based on pre-existing data that includes ratings from other entities. In the case of this project, we will be predicting movie ratings based on a dataset that contains ratings for many movies by many users.

The goal of this project is to create a **Recommendation System model** that is optimised to predict movie ratings based on the existing MovieLens 10M dataset, and takes into account various effects/biases in the dataset.

For more details on the MoveLens 10M dataset please refer to the readme file here: https://files.grouplens.org/datasets/movielens/ml-10m-README.html. The MovieLens 10M dataset (https://grouplens.org/datasets/movielens/) used in this project is provided by GroupLens:

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

To measure and compare accuracy of the models that are produced, the Root Mean Squared Error (RMSE) method is used, which provides a measure of the deviation between actual values from the dataset and predicted values from a model. The larger the RMSE, the more our predictions deviate from actual ratings. For example with an RMSE greater than 1 this means our predicted rating for a movie is more than 1 full star (rating value) away from the actual recorded ratings.

The following formula is used in this analysis to calculate the RMSE for each model:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{y}_{u,i} - y_{u,i})^2}$$

This formula is represented as the following in R code:

```
# Define RMSE algorithm to be used for evaluating prediction performance

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

1.2 Approach Summary

The approach used for this analysis consisted of the following steps, which are represented in the accompanying R script as Sections One through Eight:

1. Data sourcing and preparation - This step loads all the required libraries, acquires the dataset in zip format and extracts it, performs some preparation of columns, splits the dataset into a working dataframe called **edx** and a validation dataframe called **validation**. This step uses code already provided in the courseware.
2. Further preparation and data exploration - In this step the **edx** dataset is further split into **train** and **test** dataframes to ensure the **validation** set is not used for model creation. Some basic dataset exploration is then performed to help build awareness of the data.
3. Construct RMSE function and generate Naive analysis - This section creates the RMSE function that will be used throughout the analysis, and generates a Naive model based on the data in the **train** set that is used to generate predictions that are compared with the **test** set. Finally a tibble is constructed which will ultimately store all the RMSE results from the subsequent model analysis.
4. Consider Movie Effect - here we consider the effect (or bias) of movie ratings based on "The Movie Effect", that some movies are rated higher due to the amount of advertising performed by movie studies, or the popularity of the actors/actresses in the movie. More visible movies will naturally tend to receive more ratings.
5. Consider Movie & User Effect - here we consider the Movie Effect as in Section Four, but also consider that some users naturally rate high, and some rate low, demonstrating another form of bias in the data which needs to be considered.
6. Regularize the Movie Effect - here we seek to address the Movie Effect in the dataset by regularising the data and penalising datapoints (ratings) that are noise/outliers. We do this by adding an additional cost value into the model, which we calculate using cross-validation to find the optimal value.
7. Regularize the Movie and User Effects - similar to section six, here we use regularisation to address both the Movie and User Effects, again using cross-validation to generate a cost value for the model to address both biases. In this section we start with a wide range for the cost value, checking between 0 and 10 at 0.25 increments, and then re-run the cross-validation across a tighter range to optimise the cost factor.
8. Run predictions on Validation set for final output RMSE - in this final section we run our final optimised model on the **Validation** set to generate our final RMSE.

2 Method

In this section we discuss the process and techniques used in the analysis and model generation. Each section below aligns with a section in the corresponding R script submitted with this report.

2.1 SECTION ONE - Data sourcing and preparation

This step loads all the required libraries, acquires the dataset in zip format and extracts it, performs some preparation of columns, splits the dataset into a working dataframe called **edx** and a validation dataframe called **validation**. This step uses code already provided in the courseware.

2.2 SECTION TWO - Data exploration

In this step the **edx** dataset is further split into **train** and **test** dataframes to ensure the **validation** set is not used for model creation.

```
# Split the edx dataset into train/test for training and regularization
edx_test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
train_set <- edx[-edx_test_index,]
test_set <- edx[edx_test_index,]

# Remove entries from the test set where there is not a corresponding user or movie
# in the training set
test_set <- test_set %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

Next, we perform some basic exploration to help build awareness of the dataset.

Dataset Size

```
# Size of full edx dataset
nrow(edx)
```

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

```
# Size of the training set
nrow(train_set)
```

```
## [1] 8100048
```

```
# Size of the test set
nrow(test_set)
```

```
## [1] 899988
```

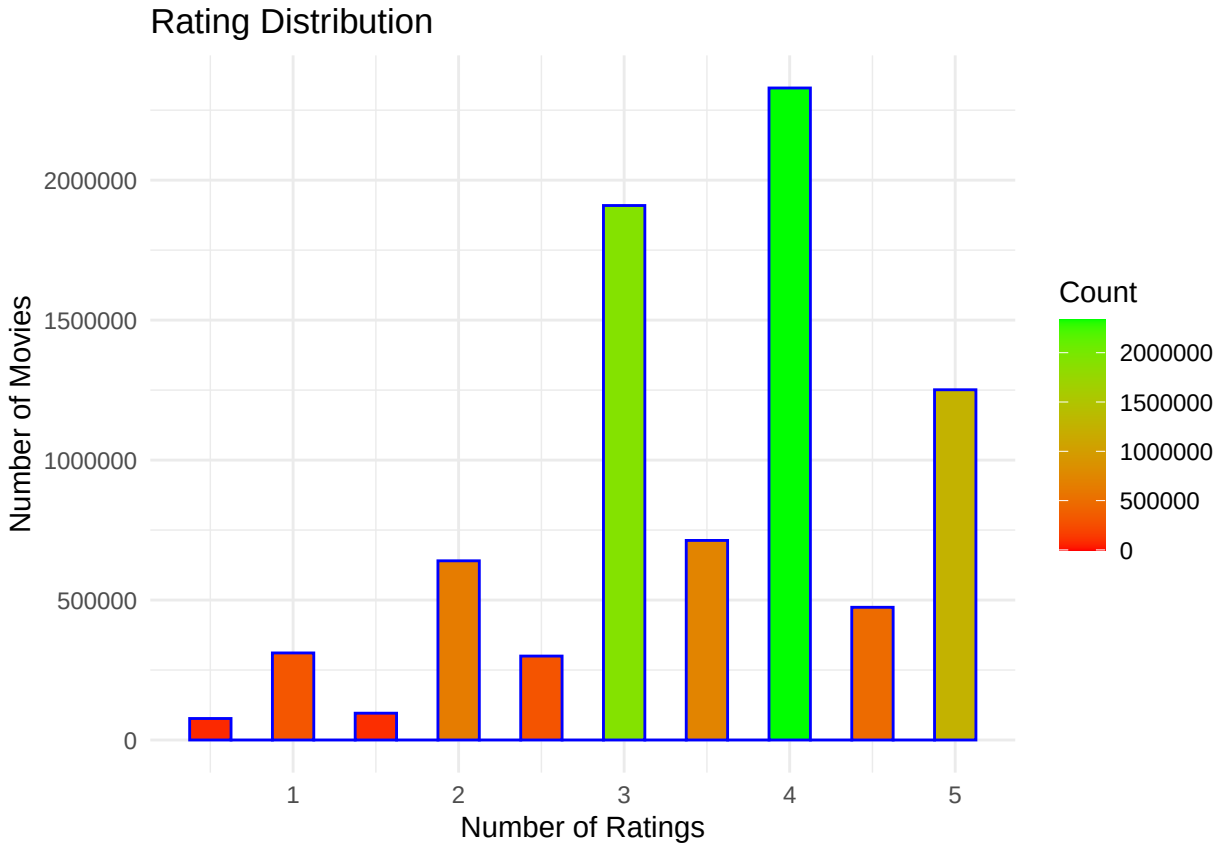
Rating Distribution

Here we see that rating values start from 0.5 rather than 0.

```
edx %>% summarize(n_rating = n_distinct(rating))
```

```
##   n_rating
## 1       10
```

We can use a basic histogram to see the distribution of average ratings for all movies:

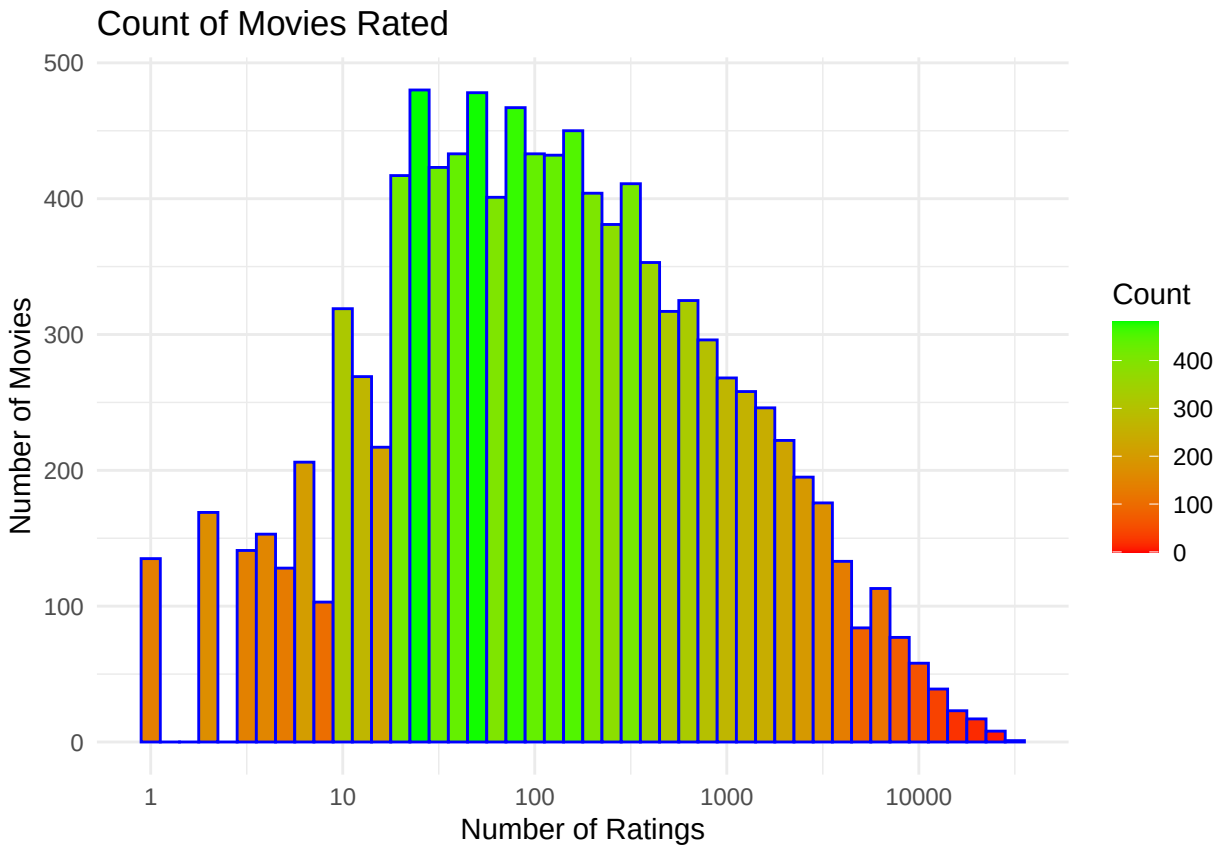


Unique users and movies

```
edx %>% summarize(n_users = n_distinct(userId),n_movies = n_distinct(movieId))
```

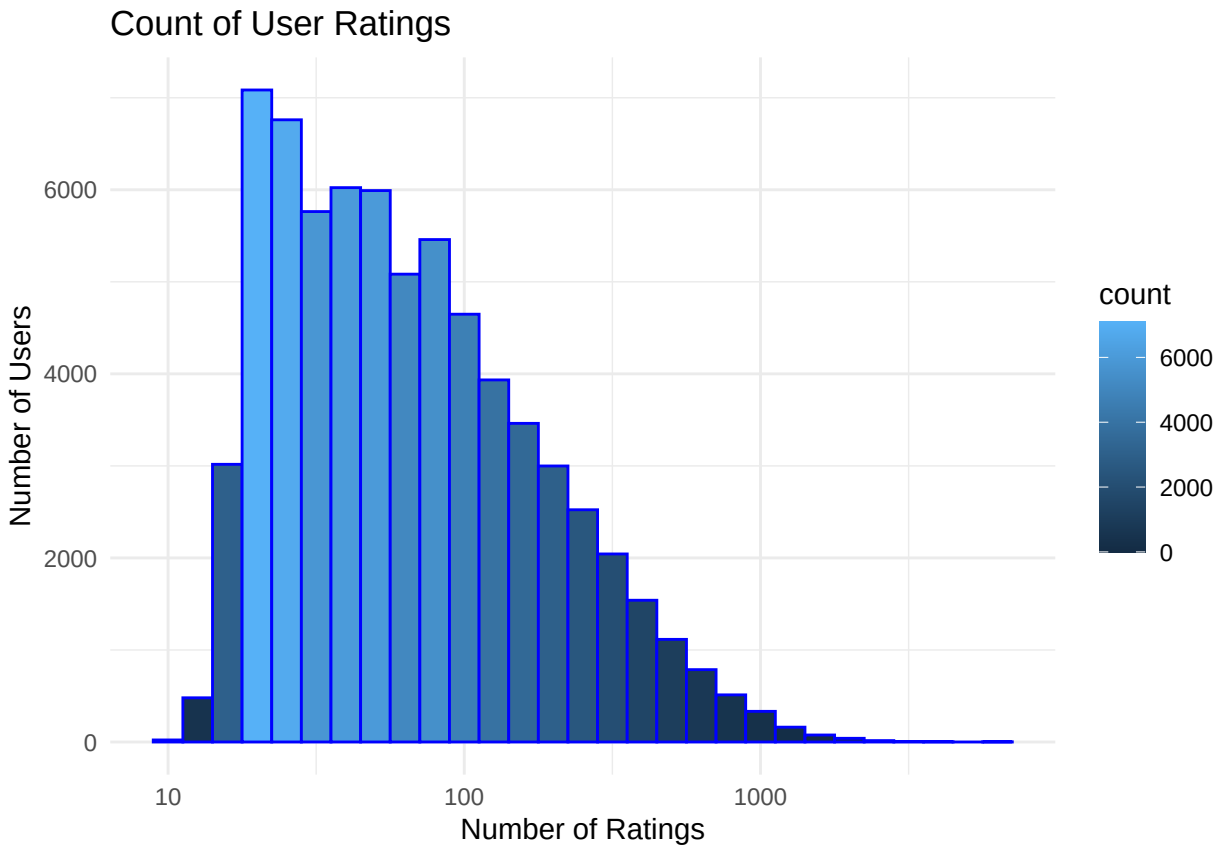
```
##   n_users n_movies
## 1   69878    10677
```

Here we can see only ~11 thousand movies were rated by ~7 thousand users, however above we can see our dataset has over 9 million data-points (rows), therefore users must be rating multiple movies each. We should seek to understand the distribution of users ratings across movies.



From the graph above we see that some movies are rated more than others by users. This could be due to factors such as blockbuster marketing budgets raising more awareness for some movies, or high profile actors/actresses appearing in the movie, ensuring more users see that movie, thereby increasing the likelihood of more users rating the movie. This represents a bias in the data that should be factored into our recommendation system model.

Additionally we will explore the rating behaviors of our users in the graph below, to see how consistently they perform ratings of movies.



Here we see the number of users rating movies is not consistent, some users are more active at rating movies than others. This could be due to a number of factors such as:

- users making a habit of rating movies
- users only willing to rate movies they feel strongly about
- users only willing to rate movies of certain genres

Regardless of the factor driving users to rate movies, we need to factor in this user-rating bias into our recommendation system model.

2.3 SECTION THREE - Generate Naive analysis

This section creates the RMSE function that will be used throughout the analysis, and generates a Naive model based on the data in the **train** set that is used to generate predictions that are compared with the **test** set. Finally a tibble is constructed which will ultimately store all the RMSE results from the subsequent model analysis.

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

mu <- mean(train_set$rating)
naive_rmse <- RMSE(test_set$rating, mu)

rmse_results <- tibble(method = "Naive RMSE", RMSE = naive_rmse)
```

Method	RMSE
Naive RMSE	1.061135

Here we can see our rating variance is higher than 1 if we only considered the mean rating when making predictions on the test set.

2.4 SECTION FOUR - Consider Movie Effect

Here we consider the effect (or bias) of movie ratings based on "The Movie Effect", that some movies are rated higher due to the amount of advertising performed by movie studies, or the popularity of the actors/actresses in the movie. More visible movies will naturally tend to receive more ratings.

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

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

movie_effect_rmse <- RMSE(test_set$rating, predicted_ratings)

rmse_results <- bind_rows(rmse_results,tibble(method ="Movie Effect RMSE", RMSE = movie_effect_rmse ))
```

Method	RMSE
Naive RMSE	1.0611350
Movie Effect RMSE	0.9441568

Taking into account the the movie bias, the mean of actual ratings minus the overall mean, we can see a slight drop in our RMSE, which is a good sign, but we can better this number by addressing other biases.

2.5 SECTION FIVE - Consider Movie & User Effect

Here we consider the Movie Effect as in Section Four, but also consider that some users naturally rate high, and some rate low, demonstrating another form of bias in the data which needs to be considered.

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

predicted_ratings <- test_set %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

movie_user_effect_rmse <- RMSE(test_set$rating, predicted_ratings)

rmse_results <- bind_rows(rmse_results,tibble(method ="Movie & User Effect RMSE", RMSE = movie_user_eff
```

Method	RMSE
Naive RMSE	1.0611350
Movie Effect RMSE	0.9441568
Movie & User Effect RMSE	0.8659736

Once again we can see a drop in RMSE by adjusting for overall user behaviour.

2.6 SECTION SIX - Regularize the Movie Effect

Here we seek to address the Movie Effect in the dataset by regularizing the data and penalizing data points (ratings) that are noise/outliers. We do this by adding an additional cost value into the model, which we calculate using cross-validation to find the optimal value.

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

reg_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(reg_sum, by='movieId') %>%
    mutate(b_i = s/(n_i+l)) %>%
    mutate(pred = mu + b_i) %>%
    pull(pred)
  return(RMSE(test_set$rating, predicted_ratings))
})

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

```
## [1] 1.75
```

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

```
## [1] 0.9441568
```

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

rmse_results <- bind_rows(rmse_results,tibble(method ="Movie Regularization Effect RMSE", RMSE = movie_
```

Method	RMSE
Naive RMSE	1.0611350
Movie Effect RMSE	0.9441568
Movie & User Effect RMSE	0.8659736
Movie Regularization Effect RMSE	0.9441568

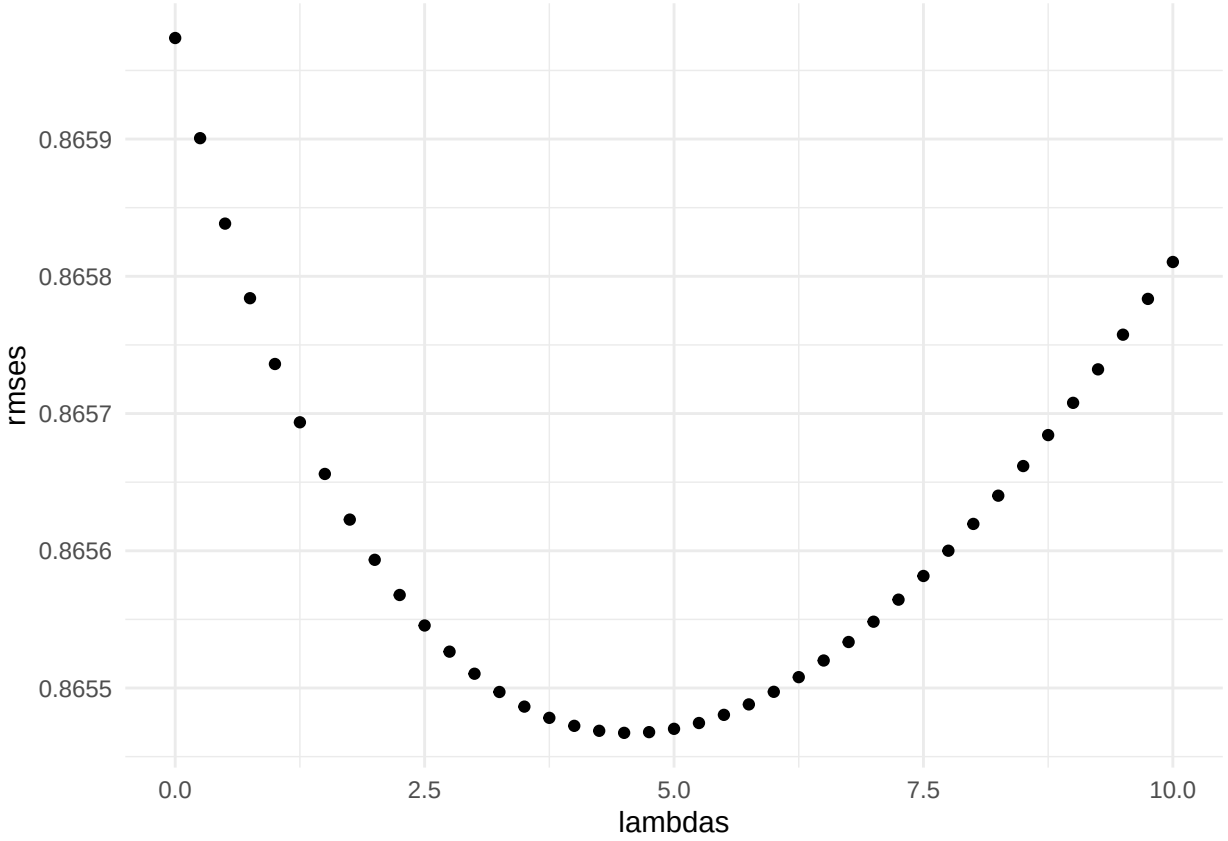
Interestingly the regularization of Movie ratings alone has a negative effect on our RMSE, clearly any outliers here don't have enough of an impact on predictability.

2.7 SECTION SEVEN - Regularize the Movie and User Effects

Similar to section six, here we use regularisation to address both the Movie and User Effects, again using cross-validation to generate a cost value for the model to address both biases. In this section we start with a wide range for the cost value, checking between 0 and 10 at 0.25 increments, and then re-run the cross-validation across a tighter range to optimize the cost factor.

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

rmses <- sapply(lambdas, function(l){

  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) %>%
    pull(pred)

  return(RMSE(predicted_ratings, test_set$rating))
})

best_wide_lambda <- lambdas[which.min(rmses)]

movie_user_reg_effect_rmse <- min(rmses)

rmse_results <- bind_rows(rmse_results,tibble(method ="Movie & User Regularization Effect RMSE", RMSE =
```

Here we can see our RMSE lowers at around 4.5 for our lambda, in the next step we try to find a tighter value for this lambda before moving to our validation set.

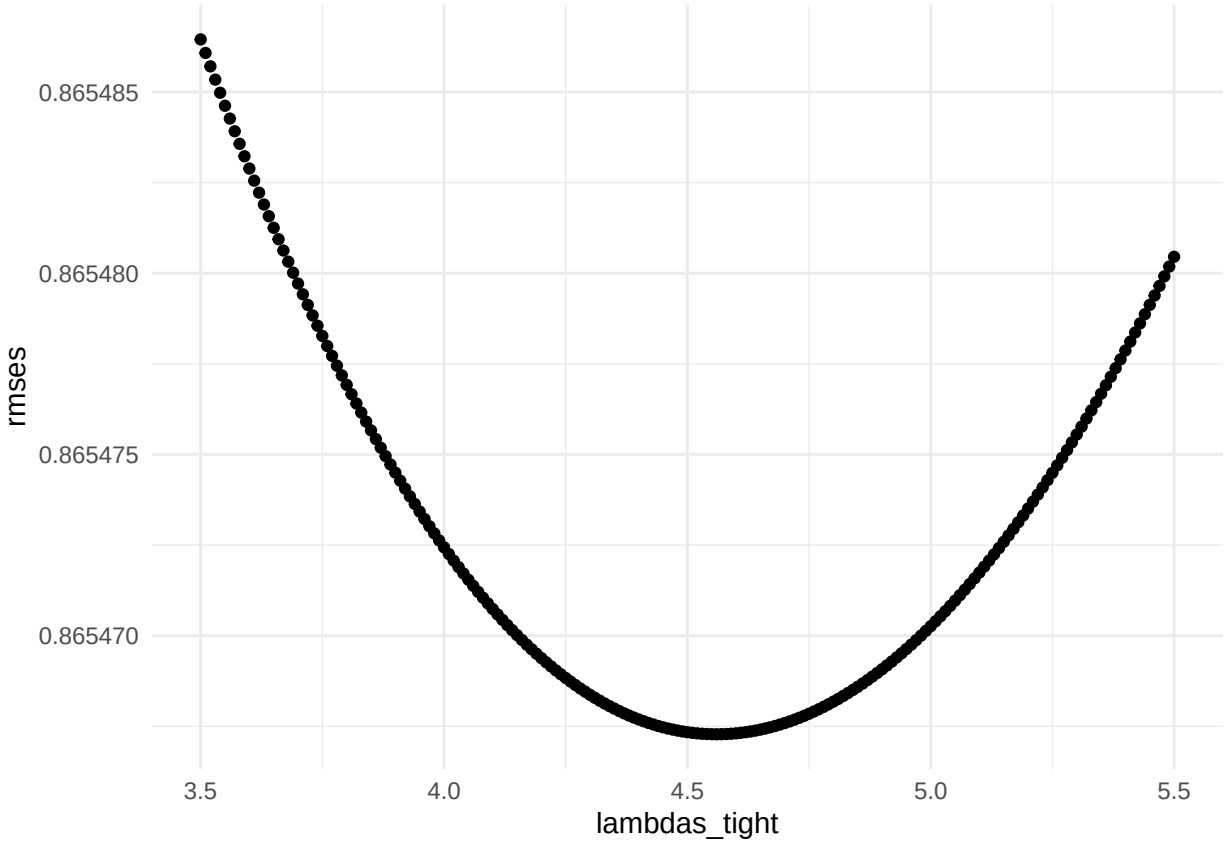


Method	RMSE
Naive RMSE	1.0611350
Movie Effect RMSE	0.9441568
Movie & User Effect RMSE	0.8659736
Movie Regularization Effect RMSE	0.9441568
Movie & User Regularization Effect RMSE	0.8654673

```
start_lambda <- best_wide_lambda - 1
end_lambda <- best_wide_lambda + 1

lambdas_tight <- seq(start_lambda, end_lambda, 0.01)

rmses <- sapply(lambdas_tight, function(l){

  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) %>%
```

```
    pull(pred)

  return(RMSE(predicted_ratings, test_set$rating))
})

best_tight_lambda <- lambdas_tight[which.min(rmses)]

movie_user_opt_reg_effect_rmse <- min(rmses)

rmse_results <- bind_rows(rmse_results,tibble(method ="Optimised Movie & User Regularization Effect RMSE
```

By narrowing our lambda range we've slightly improved the lambda value, however it has had little impact on the overall RMSE.



Method	RMSE
Naive RMSE	1.0611350
Movie Effect RMSE	0.9441568
Movie & User Effect RMSE	0.8659736
Movie Regularization Effect RMSE	0.9441568
Movie & User Regularization Effect RMSE	0.8654673
Optimised Movie & User Regularization Effect RMSE	0.8654673

By considering both Movie and User based rating outliers with the regularization we've had a marginal drop in RMSE over the non-regularized Movie & User model.

2.8 SECTION EIGHT - Validate and output final RMSE

In this final section we run our final optimized model on the **Validation** set to generate our final RMSE. We use the **best_tight_lambda** value calculated above to generate the new User and Movie bias values.

```
rmse_validation <- sapply(best_tight_lambda, function(l){

  b_i <- validation %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  b_u <- validation %>%
    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) %>%
    pull(pred)

  return(RMSE(predicted_ratings, validation$rating))
})

rmse_results <- bind_rows(rmse_results,tibble(method ="Final RMSE on Validation Set", RMSE = rmse_valida
```

Method	RMSE
Naive RMSE	1.0611350
Movie Effect RMSE	0.9441568
Movie & User Effect RMSE	0.8659736
Movie Regularization Effect RMSE	0.9441568
Movie & User Regularization Effect RMSE	0.8654673
Optimised Movie & User Regularization Effect RMSE	0.8654673
Final RMSE on Validation Set	0.8390263

3 Conclusion

```
## [1] "Final RMSE score is:  0.839026336598481"
```

Our final output when factoring in User/Movie biases, and regularizing the data, results in a final RMSE below the required value of *0.86490* for this course.

This analysis and report stepped through generating models that included various biases, and outputted a consolidated model that was run on a validation set. The biases that were assessed include:

- Movie Effect
- Movie and User Effect
- Regularized Movie Effect (Note: this was surprisingly not effective, indicating User Effect regularization was more important in this dataset.)
- Regularized Movie and User Effect

The analysis showed that by factoring in these biases our model gradually improved it's RMSE, demonstrating increased accuracy of predictions.

The final model produced an RMSE on the Validation set that met the measure expected under the assignment. This value could potentially be lowered by considering other forms of bias. Some users tend to rate only their favorite movie genres, thereby increasing the number of ratings over a particular pool of movies. Time could also be a factor, where the number of ratings of movies in the dataset increase over time as rating systems became more accessible over the internet. The data to assess these biases are available in the dataset, but have not been considered as we have already achieved a successful RMSE.