Final_Project_MovieLens

Alfredo Nasiff

3/23/2020

Introduction

The Movie Lens project is part of the HarvardX: PH125.9x Data Science: Capstone course. It closely follows the Data Science textbook by Rafael Irizarry paricularly; Chapter 33.7 Recommendation systems, and Chapter 33.9 Regularization. The dataset was supplied as part of the course.

The aim of the project is to develop and train a recommendation machine learning algorithm to predict a rating given by a users to a movie in the dataset. The Residual Mean Square Error (RMSE) will be used to evaluate the accuracy of the algorithm. The required criteria for the projects is a RMSE lower than 0.86490.

This report will present an overview of the data, analysis, results and a conclusion.

Dataset

An introductory review of the dataset is performed in order to familiarise ourselves. Data is downloaded as per instruction from the MovieLens 10M dataset.

```
##########################################################################
## Create edx set, validation set
##########################################################################
# Note: this process could take a couple of minutes
# Package Installs
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")
# Libraries
library(tidyverse)
library(caret)
library(hexbin)
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip
# Check for download
datafile <- "MovieLens.RData"
if(!file.exists("MovieLens.RData"))
{
  print("Download")
  dl <- tempfile()
  download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
  ratings <- read.table(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") # if using R 3.6.0: 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,]

  # To make sure we don't include users and movies in the test set that do not appear in the training (

  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)
  edx <- rbind(edx, removed)
  rm(dl, ratings, movies, test_index, temp, movielens, removed)
  save(edx, validation, file = datafile)
} else {
  load(datafile)
}
```

Edx dataset contains rows corresponding to users' ratings of a movie. The set contains the variables; "userId", "movieId", "rating", "timestamp", "title", "genres".

```
# Summarise Data
head(edx, 5)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
##                           genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5        Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
```

Summarising the dataset reveals a well formatted set with no missing values. Movies are rated between 0.5 and 5.0, with 9000055 rows in total.

```
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:9000055     Length:9000055
##  Class :character   Class :character
```

```
##  Mode   :character   Mode   :character
##
##
##
```

The dataset contains ~10,700 unique movies, ~70,000 unique movies, and ~800 unique combinations of genres, and a mean movie rating of ~3.5 out of 5.

```
# Movies, Users and Genres in Database
edx %>% summarise(
  uniq_movies = n_distinct(movieId),
  uniq_users = n_distinct(userId),
  uniq_genres = n_distinct(genres))
```

```
##   uniq_movies uniq_users uniq_genres
## 1       10677      69878         797
```

```
# Ratings Mean
rating_mean <- mean(edx$rating)
rating_mean
```

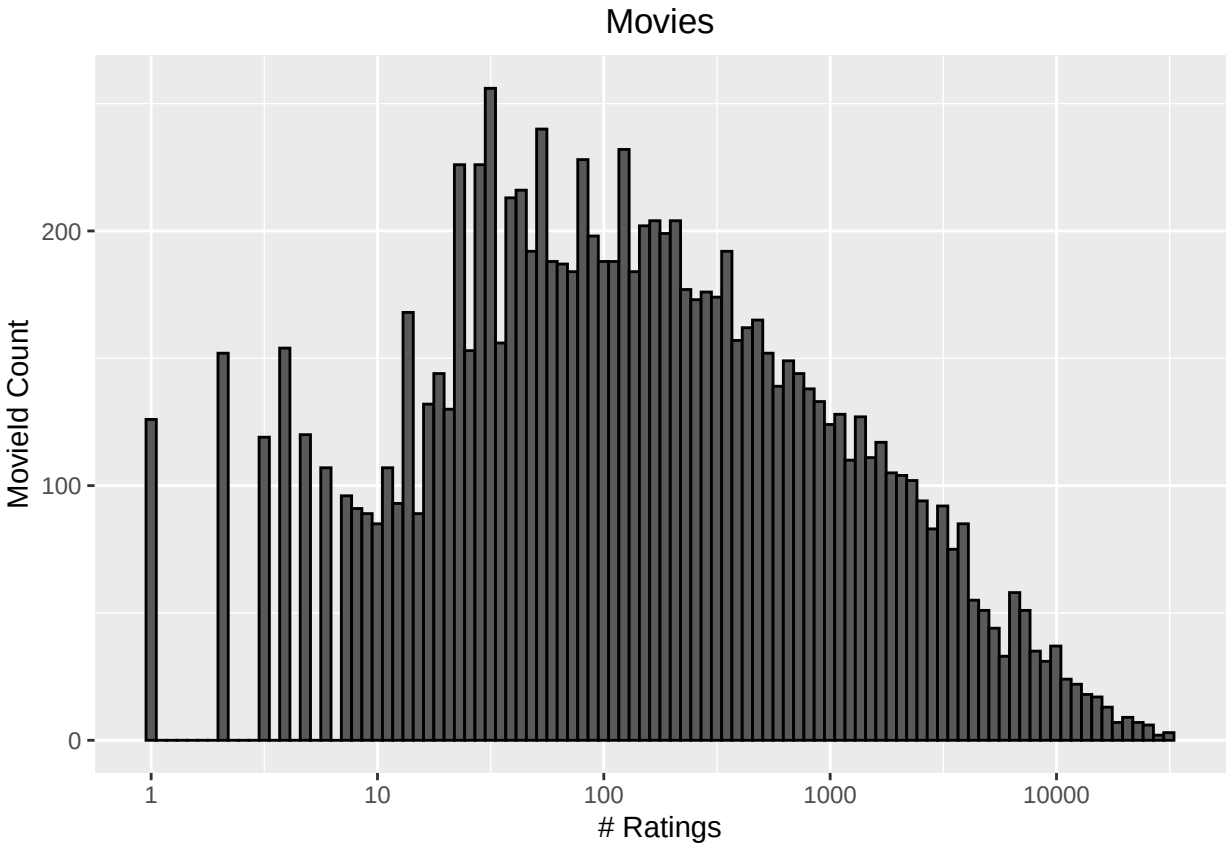
```
## [1] 3.512465
```

In this machine learning challenge each outcome Y has a different set of predictors. To see this, note that if we are predicting the rating for movie i by user u, in principle, all other ratings related to movie i and by user u may be used as predictors, but different users rate different movies and a different number of movies. Furthermore, we may be able to use information from other movies that we have determined are similar to movie i or from users determined to be similar to user u. In essence, the entire matrix can be used as predictors for each cell.

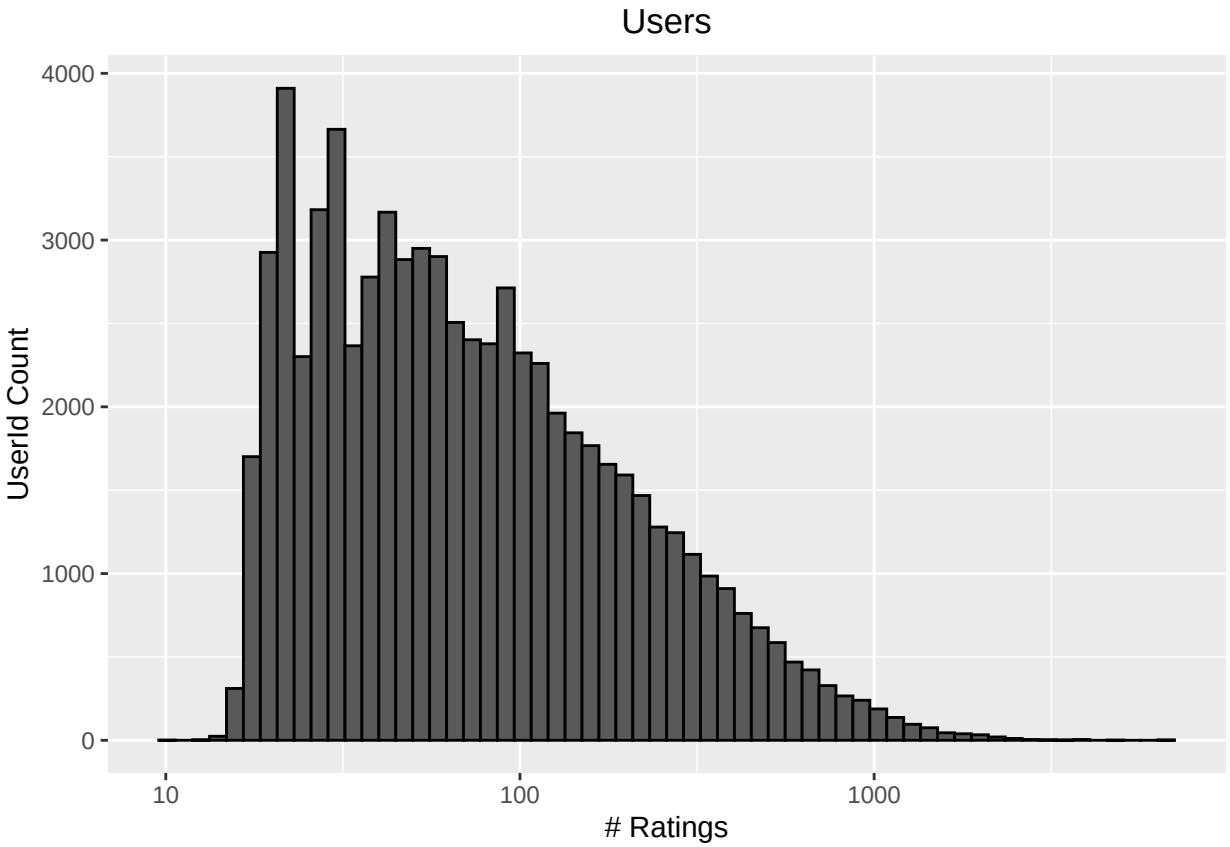
Let's look at some of the general properties of the data to better understand the challenges.

The first thing we notice is that some movies get rated more than others. Below is the distribution. This should not surprise us given that there are blockbuster movies watched by millions and artsy, independent movies watched by just a few. Our second observation is that some users are more active than others at rating movies:

```
# Ratings Users - Number of Ratings

edx %>% count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(color = "black", bins=100) +
  scale_x_log10() +
  xlab("# Ratings") +
  ylab("MovieId Count") +
  ggtitle("Movies") +
  theme(plot.title = element_text(hjust = 0.5))
```




```
edx %>% count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(color = "black", bins=60) +
  scale_x_log10() +
  xlab("# Ratings") +
  ylab("UserId Count") +
  ggtitle("Users") +
  theme(plot.title = element_text(hjust = 0.5))
```


Users
4000
3000
2000
1000
0
UserId Count
10
100
1000
Ratings

Analysis and Results

The Residual Mean Square Error (RMSE) is the error function to that will measure accuracy and quantify the typical error we make when predicting the movie rating. An error larger than 0.8775, it means our typical error is larger than the required for this assignment almost a star, which is not good. RMSE defined;

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

where; N is the number of users, movie ratings, and the sum incorporating the total combinations.

Simple Prediction Based on Mean Rating

We can use a model based approach to answer this. A model that assumes the same rating for all movies and users with all the differences explained by random variation would look like this:

$$Y_{u,i} = \mu + \epsilon_{u,i}$$

where $Y_{u,i}$ is the prediction, with $\epsilon_{u,i}$ the independent error sampled from the same distribution centered at 0 and μ, the expected "true" rating for all movies. We know that the estimate that minimizes the RMSE is the least squares estimate of μ and, in this case, is the average of all ratings:

```
## Simple Prediction based on Mean Rating
mu <- mean(edx$rating)
mu
```

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

If we predict all unknown ratings with μ we obtain the following RMSE:

```
rmse_naive <- RMSE(validation$rating, mu)
rmse_naive
```

```
## [1] 1.061202
```

```
## Save Results in Data Frame
rmse_results = tibble(method = "Naive Analysis by Mean", RMSE = rmse_naive)
rmse_results %>% knitr::kable()
```

method	RMSE
Naive Analysis by Mean	1.061202

Investigating the dataset allows for more advanced analysis and rating predictions with smaller error.

Movie Effects Model

We know from experience that some movies are just generally rated higher than others. This intuition, that different movies are rated differently, is confirmed by data. We can augment our previous model by adding

the term b_i to represent average ranking for movie i:

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i}$$

where b_i is the bias for each movie i.

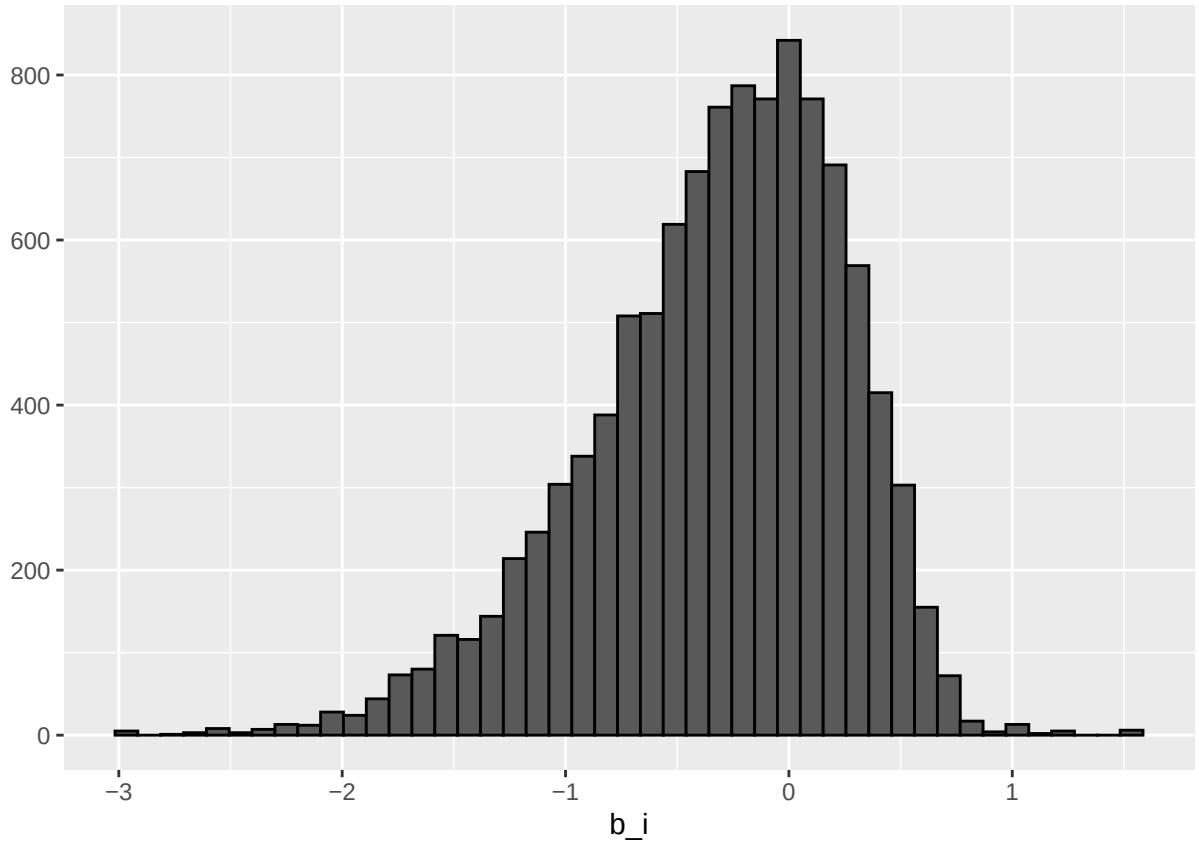
```
## Simple model taking into account the movie effects, b_i
mu <- mean(edx$rating)
mu
```

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

```
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating - mu))
```

We can see that these estimates vary substantially:

```
qplot(b_i, data = movie_avgs, bins = 45, color = I("black"))
```



Remember that $\hat{\mu}$=3.5, so a b_i=1.5 implies a perfect five star rating. Let's see how much our prediction improves once we use $\hat{Y}_{u,i} = \hat{\mu} + \hat{b_i}$:

```
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
rmse_model_movie_effects <- RMSE(predicted_ratings, validation$rating)
rmse_model_movie_effects
```

```
## [1] 0.9439087
```

```
## Add Results in Data Frame
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie Effects Model",
                          RMSE = rmse_model_movie_effects))
rmse_results %>% knitr::kable()
```

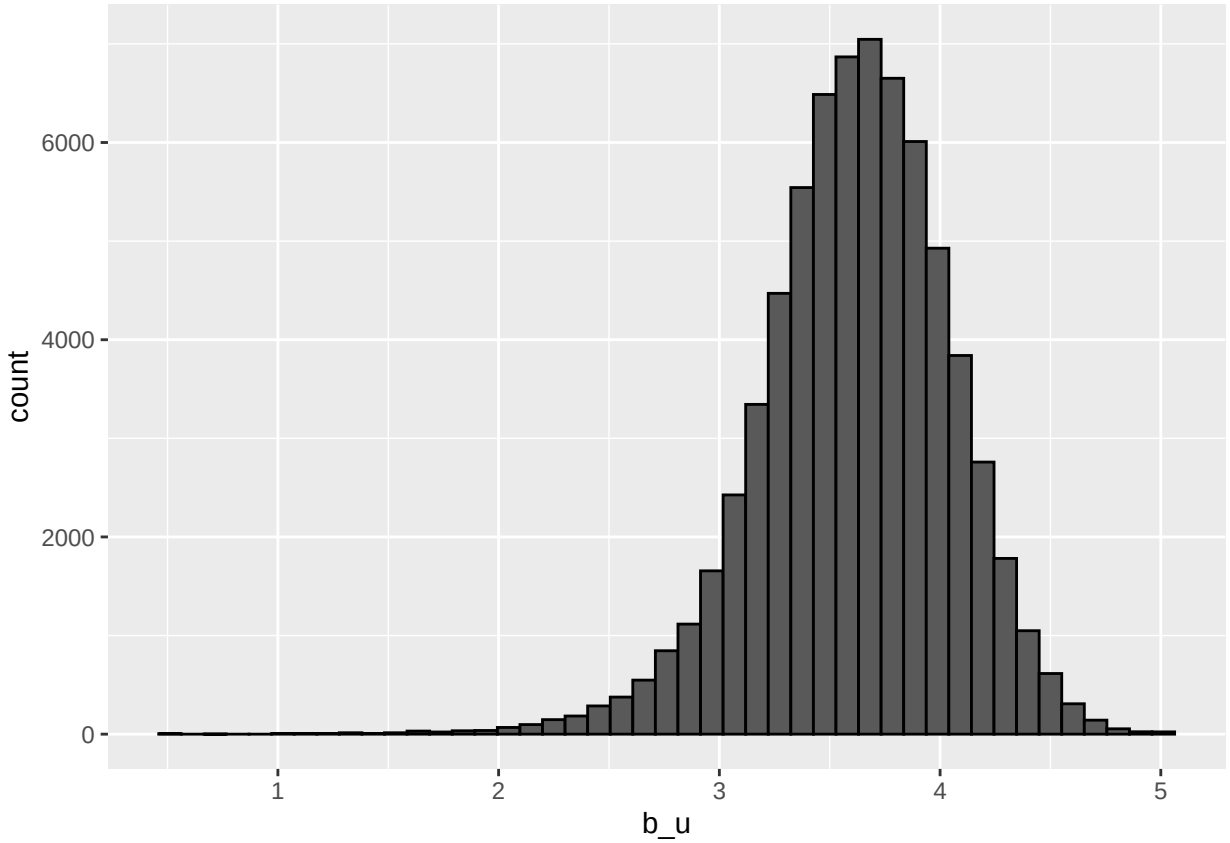
method	RMSE
Naive Analysis by Mean	1.0612018
Movie Effects Model	0.9439087

The Movie Effect Model; predicting the movie rating with both bias, b_i, and mean, μ gives an improved prediction with a lower RMSE value.

Movie and User Effects Model

The next step is to incorporate the individual User Effects, b_u, in to the model. Acknowledging each user inherent bias to mark all films higher or lower. Let's compute the average rating for user u for those that have rated over 100 movies:

```
edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 45, color = "black")
```



Notice that there is substantial variability across users as well: some users are very cranky and others love every movie. This implies that a further improvement to our model may be:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

where b_u is the bias for each user u. Now if a cranky user (negative b_u) rates a great movie (positive b_i), the effects counter each other and we may be able to correctly predict that this user gave this great movie a 3 rather than a 5. We will compute an approximation by computing μ and b_i and estimating b_u as the average of $Y_{u,i} - \hat{\mu} - \hat{b_i}$:

```
## Movie and User Effects Model
# Simple model taking into account the user effects, b_u
user_avgs <- edx %>%
  left_join(movie_avgs, by="movieId") %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu - b_i))
```

We can now construct predictors and see how much the RMSE improves:

```
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
rmse_model_user_effects <- RMSE(predicted_ratings, validation$rating)
rmse_model_user_effects
```

```
## [1] 0.8653488
```

```
rmse_results <- bind_rows(rmse_results,
                          data_frame(method="Movie and User Effects Model",
                                     RMSE = rmse_model_user_effects))
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

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

method	RMSE
Naive Analysis by Mean	1.0612018
Movie Effects Model	0.9439087
Movie and User Effects Model	0.8653488

Incorporating the user bias into the model resulted in a further reduced RMSE.

Regularization

Regularization allows for reduced errors caused by movies with few ratings which can influence the prediction and skew the error metric. The method uses a tuning parmeter, λ, to minimise the RMSE. Modifying b_i and b_u for movies with limited ratings.

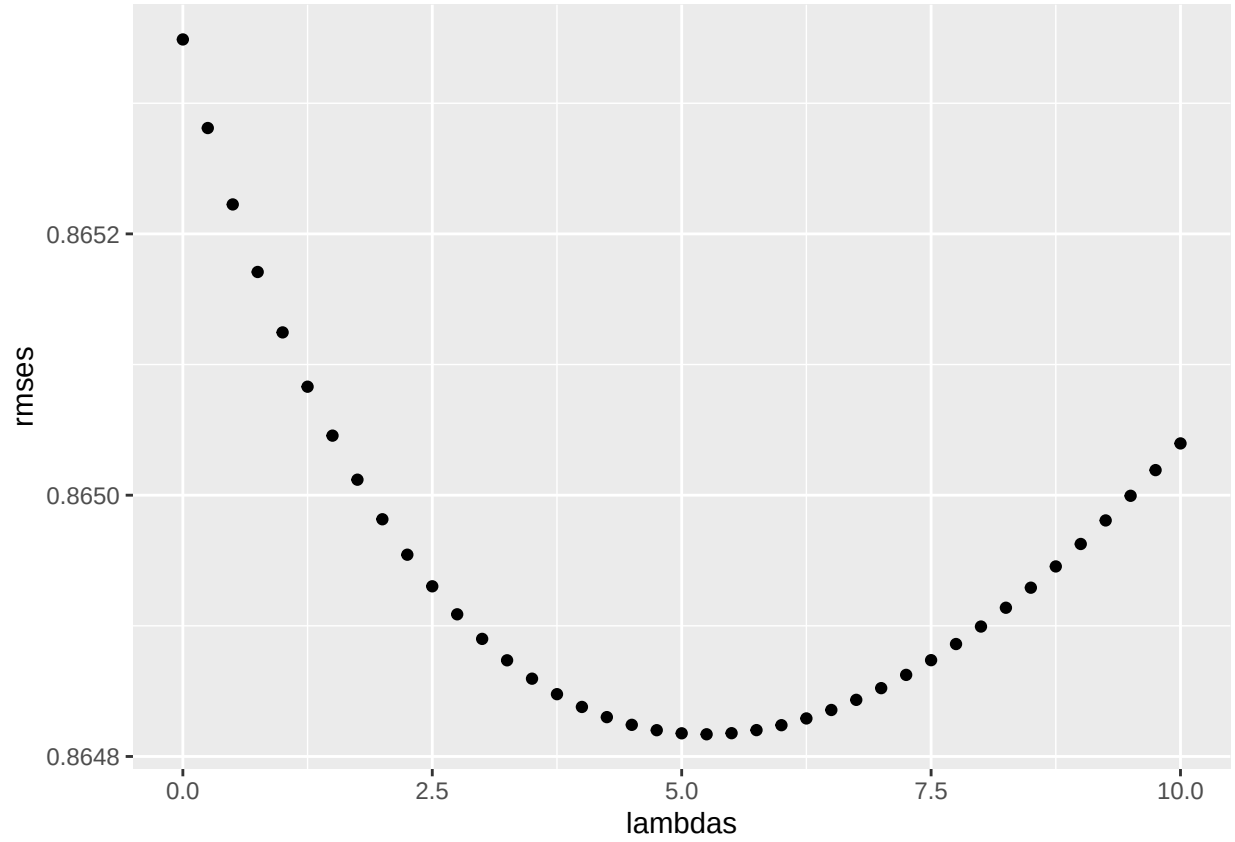
$$Y_{u,i} = \mu + b_{i,n,\lambda} + b_{u,n,\lambda} + \epsilon_{u,i}$$

```
# Predict via regularization, movie and user effect model
# (as per https://rafalab.github.io/dsbook 34.9 Regularization)
lambdas <- seq(0, 10, 0.25)
rmses <- sapply(lambdas, function(l){

  mu <- mean(edx$rating)

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

  b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarise(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_regularization <- min(rmses)
rmse_regularization
```

```
## [1] 0.864817
```

```
# Plot RMSE against Lambdas to find optimal lambda
qplot(lambdas, rmses)
```




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

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

```
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Regularized Movie and User Effects Model",
                                 RMSE = rmse_regularization))
rmse_results %>% knitr::kable()
```

method	RMSE
Naive Analysis by Mean	1.0612018
Movie Effects Model	0.9439087
Movie and User Effects Model	0.8653488
Regularized Movie and User Effects Model	0.8648170

Regularization of a Movie and User Effect model has lead to a lowest RMSE of the prediction methods for the MovieLens ratings system.

Results and Discussion

The final values of the prediction models are shown below;

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

method	RMSE
Naive Analysis by Mean	1.0612018
Movie Effects Model	0.9439087
Movie and User Effects Model	0.8653488
Regularized Movie and User Effects Model	0.8648170

The models from most accurate to least accurate are as follows; regularized Movie and User Effects Model; Movie and User Effects Model; Movie Effects Model; and Simple Average Model.

The final model optimised for the prediction is the following;

$$Y_{u,i} = \mu + b_{i,n,\lambda} + b_{u,n,\lambda} + \epsilon_{u,i}$$

The lowest value of RMSE predicted is 0.8648170.

Conclusion

A machine learning algorithm to predict the ratings from the Movie Lens dataset was constructed. The optimal model incorporated the effects of both user and movie bias in the model and these variables were regularized to eliminate movies with a low number of ratings.

The aim of the project was to develop an algorithm with lower RMSE than 0.87750, which was achieved by the Movie and User Effects and Regularized Movie and User Effects model.