MovieLens Capstone Project

Megan Lambert

11/11/2019

Executive Summary

Introduction

The goal of this project is to train a machine learning algorithm to create a movie recommendation system based on the MovieLens dataset. Many companies allow their customers to rate products so they can collect data sets and use them to predict a user's rating of a specific product. Those predictions can be used to recommend products to users that they are likely to give a high rating.

In 2006, Netflix challenged the data science community to improve their recommendation system by 10%. They use a recommendation system to predict the number of stars a user will give a certain movie. You can visit this link to learn more about the winning algorithm: https://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf

Goal

The objective of this project is to use the inputs in one subset to train a machine learning algorithm that will predict movie ratings in the validation set. Our algorithm will be evaluated based on the Residual Mean Square Error (RMSE). The RMSE is the typical error made in the predicted movie ratings. For our model, we must explore which predictors improve the RMSE. Adding additional predictors that are highly correlated with each other will not improve our model significantly. Our goal is find the model that gives us the lowest RMSE. This is the formula for RMSE:

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

Dataset Description

Since the Netflix data is not public, we will use data from GroupLens research lab's database with over 20 million ratings for over 27,000 movies by more than 138,000 users. We will use the 10M version of the MovieLens dataset, which can be found here:
https://grouplens.org/datasets/movielens/10m/

In this dataset, each row contains a rating given by one user to one movie. Not every user has rated every movie. Our dataset has the following variables: userId, movieId, rating, timestamp, title, genres. I will describe the dataset further when I go over the data exploration techniques used in the following section.

Summary of Steps Taken

Data Ingestion and Preparing

Our first step was to use the code provided by the course to download the 10M Movielens data. With this code, we also created train and validation sets to use for our models.

Data Exploration and Visualization

Next we explored the dataset and its variables. We also plotted some of the variables to look for potential predictors for our models.

Defining Our Performance Measure

We defined our performance measure as the model with the lowest RMSE. We defined the RMSE function for use in our models.

Comparing Models

We compared models that incorporated movie and user effects on ratings.

Regularization

We used cross validation to choose the optimal lamda, which is our tuning parameter. Then we made a regularized model for movie and user effects, which obtained the lowest RMSE of 0.8648170

Methods and Analysis

Downloading the Dataset

The first step was to run the code provided by the course module to download the dataset:

```r
################################
# Create edx set, validation set
################################

# Note: this process could take a couple of minutes
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")
```

Create Train and Validation Sets

The next step is to create train and validation sets using the code provided in the course module:

```
# 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")
# 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)
```

Data exploration

Now we will explore the edx dataset so we can get an idea of which variables might be good predictors to use for our model.

```
#Examine the first rows of the edx dataset with headers
head(edx)
```

```
##   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)
## 7      1     355      5 838984474       Flintstones, The (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
## 7       Children|Comedy|Fantasy
```

```
#View the summary statistics for the edx dataset
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
##
##
##
```

Here we can see the number of unique users and the number of unique movies:

```r
#Number of unique users and movies
edx %>% summarize(n_users=n_distinct(userId), n_movies=n_distinct(movieId))
```

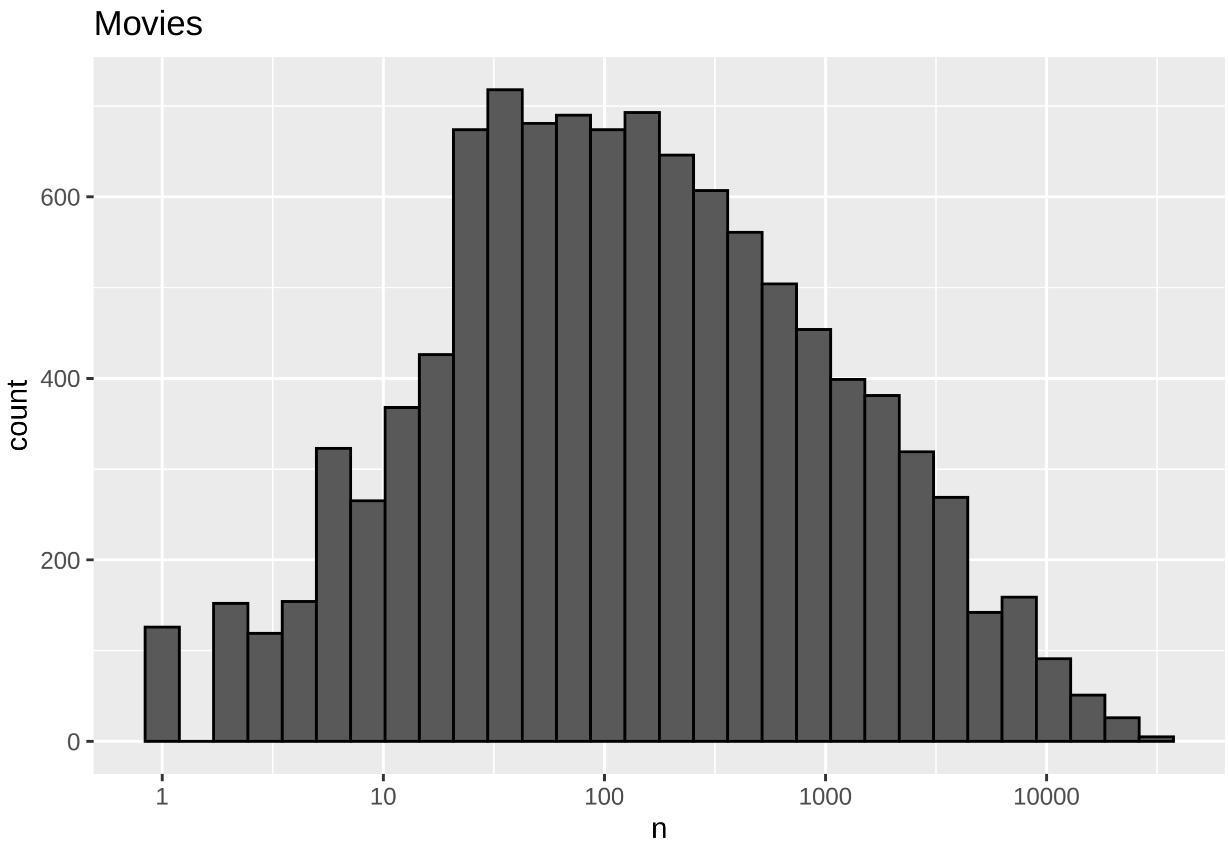
```
##   n_users n_movies
## 1   69878    10677
```

```r
#Number of rows in edx dataset
nrow(edx)
```

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

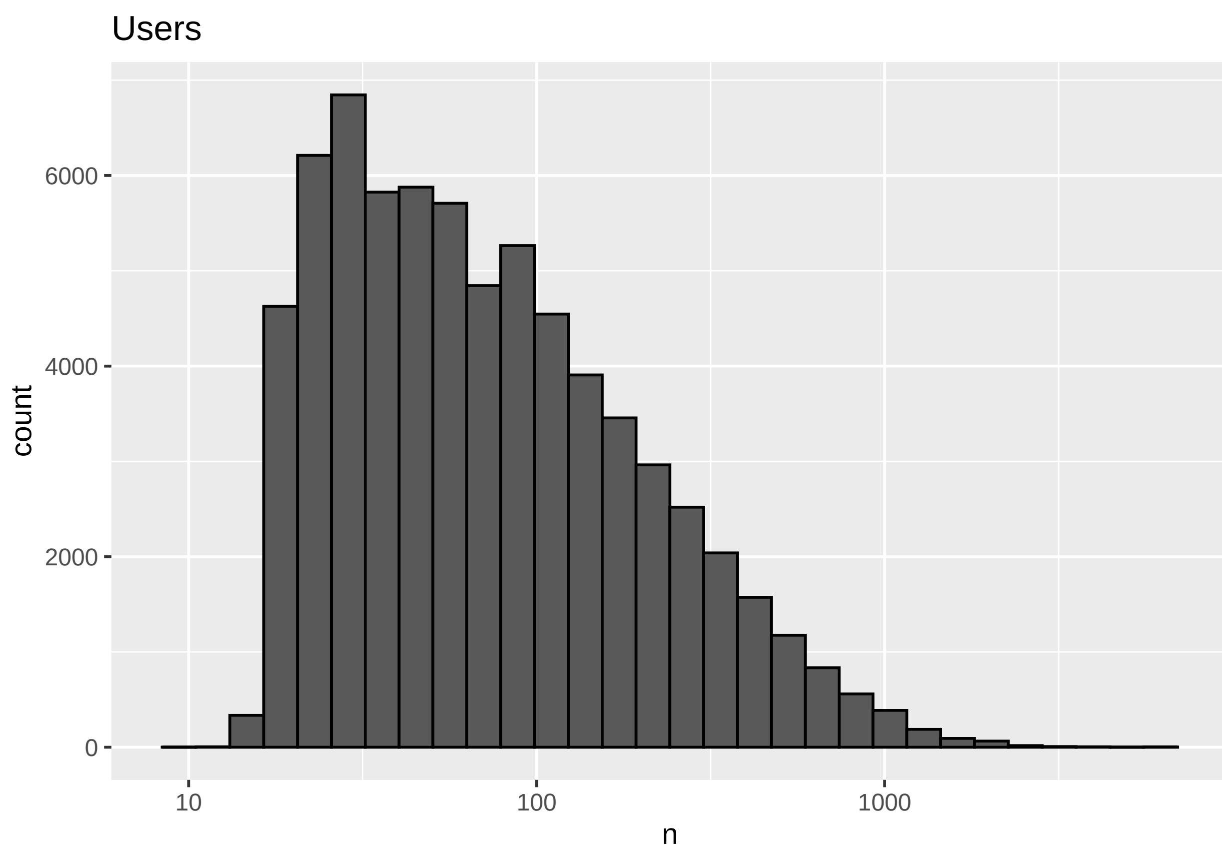
Data Visualization

```r
#Visualize the fact that some movies get rated more times than others
edx %>%     dplyr::count(movieId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 30, color = "black") +
scale_x_log10() +
ggtitle("Movies")
```



Most users have rated more than 30 movies.

```
#Visualize the fact that some users rate more movies than others
edx %>%
     dplyr::count(userId) %>%
     ggplot(aes(n)) +
     geom_histogram(bins = 30, color = "black") +
     scale_x_log10() +
     ggtitle("Users")
```



By analyzing the number of ratings for each score, we can see that whole number ratings are more popular than half number ratings.

```
#Looking at how the number of ratings are distributed
edx %>% group_by(rating) %>% summarize(count = n())
```

```
## # A tibble: 10 x 2
##    rating   count
##     <dbl>   <int>
##  1    0.5   85374
##  2    1    345679
##  3    1.5  106426
##  4    2    711422
##  5    2.5  333010
##  6    3   2121240
##  7    3.5  791624
##  8    4   2588430
##  9    4.5  526736
## 10    5   1390114
```

RMSE function

We define the RMSE function that we will use as a measure of performance. The model with the lowest RMSE will be the best model because it gives the lowest errors between true ratings and predicted ratings.

```r
#Function we will use to calculate the RMSE of each model
RMSE <- function(true_ratings, predicted_ratings){
     sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Models

Our first model assumes that all movies and users have the same rating and that the differences are explained by random variation.

Mean Rating

```r
#Compute the average movie rating on the training data
mu_hat <- mean(edx$rating)
mu_hat
```

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

This is the average rating of the movies across all users.

```r
#Predict the unknown ratings to be mu and compute the residual mean squared error on the test data.
naive_rmse <- RMSE(validation$rating, mu_hat)
naive_rmse
```

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

Comparing the results of different approaches

```r
#Create a results table to display the RMSE results of our various models
rmse_results <- tibble(method = "Mean Rating Only", RMSE = naive_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Mean Rating Only	1.061202

Movie effect model

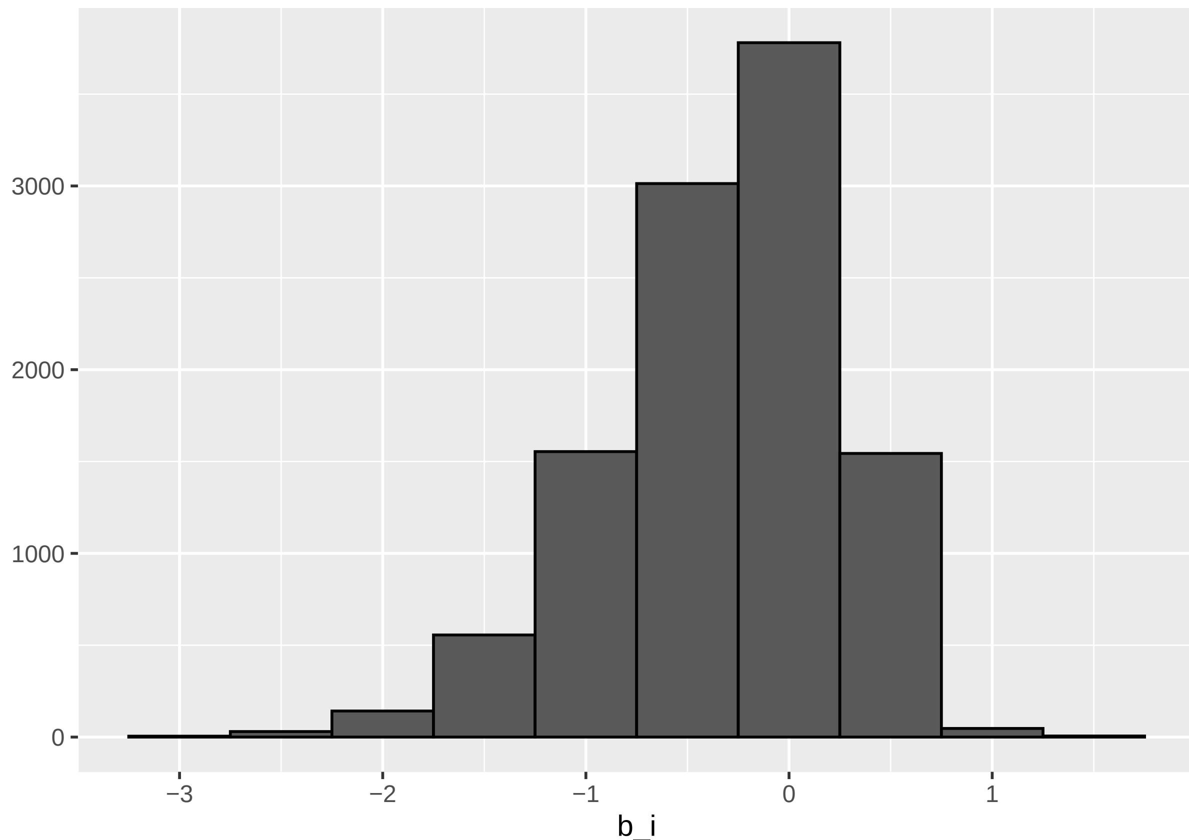
Some movies are more popular among users and tend be rated higher, so we will account for this in our next model. Here we will plot the average rating of each movie to see if the movie effect has an impact.

```r
#Define the overall movie rating mean
mu <- mean(edx$rating)
```

```r
#Compute the average rating for each movie
movie_average_rating <- edx %>%
```

```r
group_by(movieId) %>%
summarize(b_i = mean(rating-mu))
```

```r
#Plot the average movie rating
movie_average_rating %>%
  qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("black"))
```



We can see from the plot that movies are rated differently so the movie effect should be taken into account for our model.

```r
#Predict the ratings using the movie effect model
predicted_ratings_movie_effect <- mu  + validation %>%
left_join(movie_average_rating, by='movieId') %>%
.$b_i
```

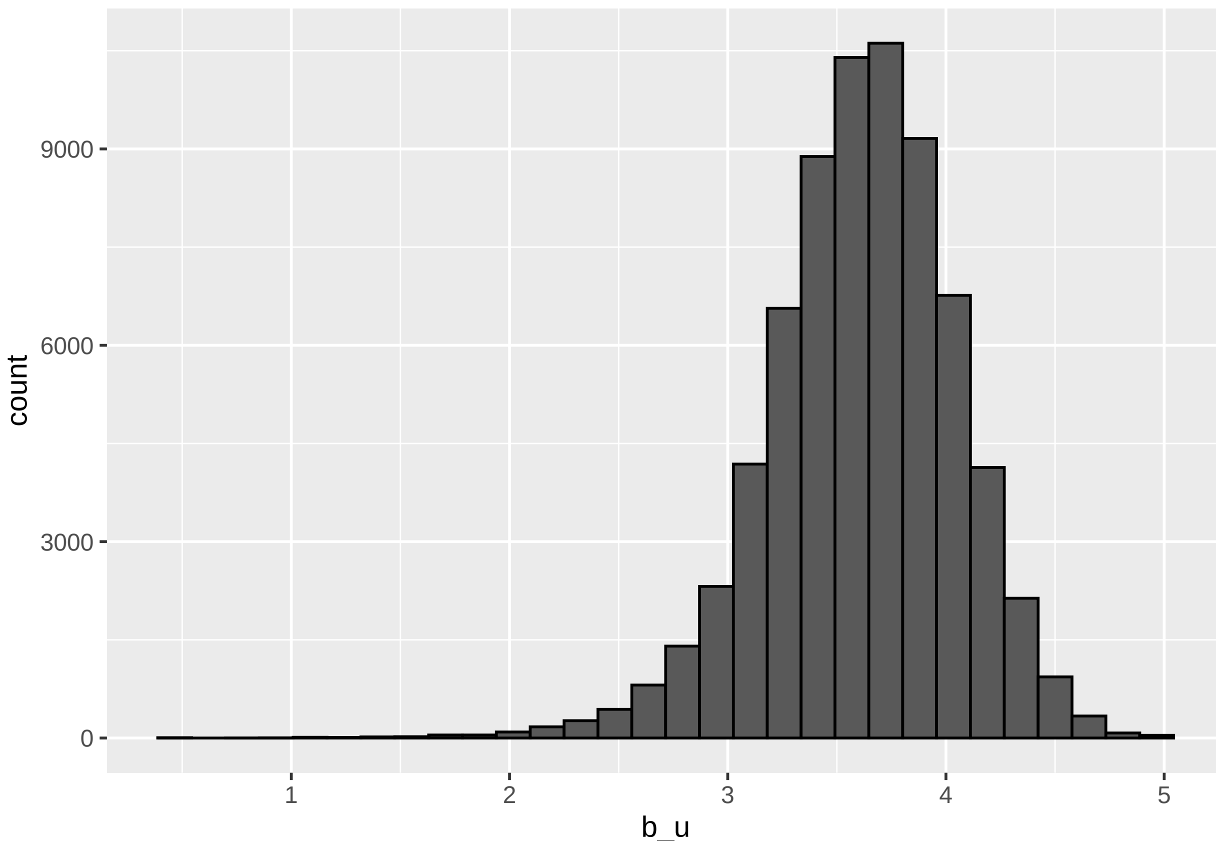
```r
#Compute the RMSE of our new movie effect model
model_movie_effect_rmse <- RMSE(predicted_ratings_movie_effect, validation$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie Effect Model", RMSE = model_movie_effect_rmse))
rmse_results %>% knitr::kable()
```

method	RMSE
Mean Rating Only	1.0612018
Movie Effect Model	0.9439087

Movie and User Effect Model

Some users tend to be more critical and have different movie preferences than others. We will add the user effect to see if that improves our model.

```
#Plot user average rating
edx %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```



From this plot we can see significant variability across users. Some users love all the movies they watch and others are more critical. We will take user bias effect into account for our model.

We will compute our approximation by calculating the overall mean, u, the movie effect, b_i, and then estimating the user effects, b_u.

```
#Estimate the user effects, b_u.
user_average_rating <- edx %>%
left_join(movie_average_rating, by='movieId') %>%
     group_by(userId) %>%
     summarize(b_u = mean(rating - mu - b_i))
```

```r
#Predict ratings using both movie and user effect
predicted_ratings_movie_and_user_effect <- validation %>%
left_join(movie_average_rating, by='movieId') %>%
left_join(user_average_rating, by='userId') %>%
mutate(pred = mu + b_i + b_u) %>%
pull(pred)
```

```r
#Compute the RMSE of the movie and user effect model
model_movie_and_user_effect_rmse <- RMSE(validation$rating,predicted_ratings_movie_and_user_effect)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Movie and User Effect Model",
                                 RMSE = model_movie_and_user_effect_rmse ))
rmse_results %>% knitr::kable()
```

method	RMSE
Mean Rating Only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488

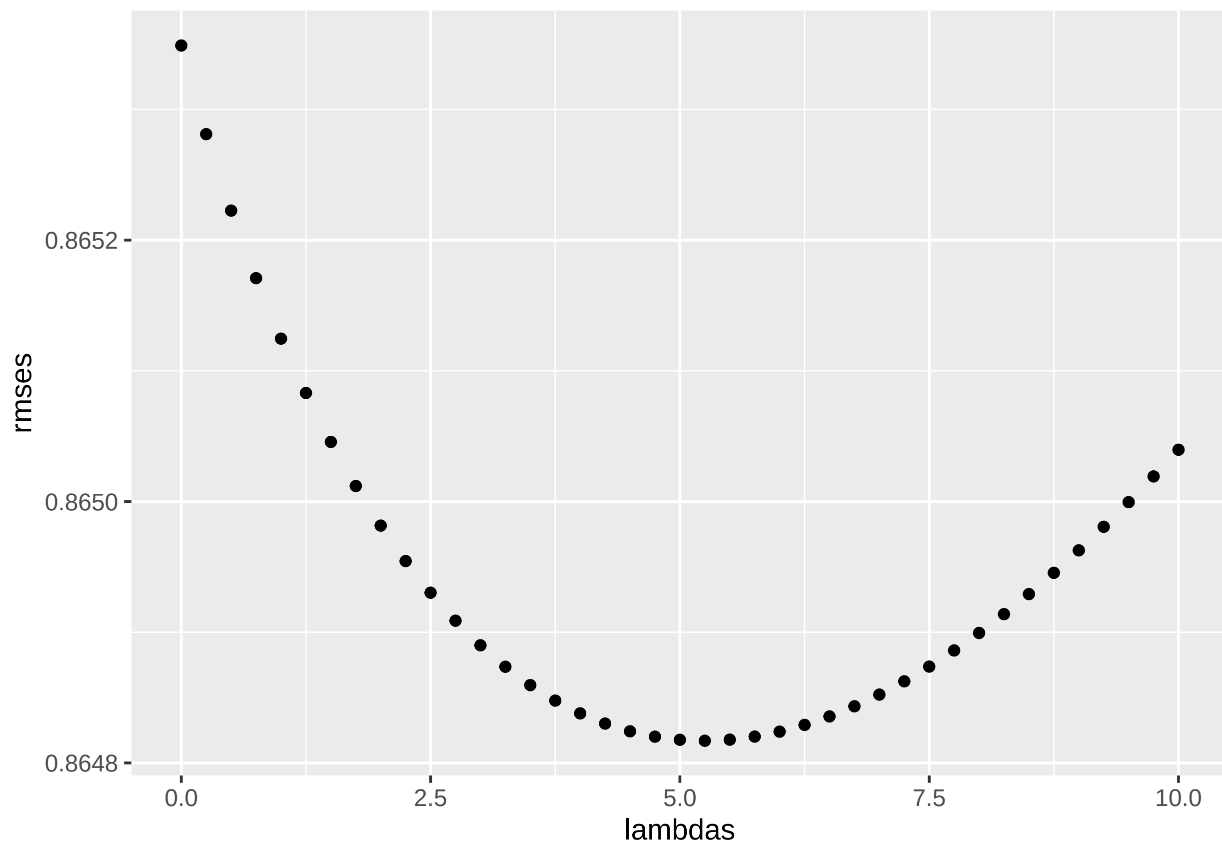
Regularization

We can further improve our results using regularization, which penalizes large estimates that come from small sample sizes in order to limit the total variability of the effect sizes. From our initial data visualization and exploration, we discovered that some movies have only been rated a few times and some users have only given a few ratings. Regularization can help reduce the risk of overfitting by accounting for the effects of movies and users with a low number of ratings. We will use lamda as our tuning parameter.

```r
#We apply a cross validation method to choose the best lamda.
lambdas <- seq(0, 10, 0.25)
#For each lamda we will calculate b_i and b_u to set our predictions and test them
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_regularized <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
```

```r
    return(RMSE(predicted_ratings_regularized, validation$rating))
})
```

```r
#Plot the lamdas and RMSEs
qplot(lambdas, rmses)
```




```r
#Find the optimal value for lamda
lambda <- lambdas[which.min(rmses)]
lambda
```

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

```r
#Compute the RMSE for the regularized Moive and User effect model
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Regularized Movie + User Effect Model",
                                 RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

method	RMSE
Mean Rating Only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie + User Effect Model	0.8648170

Results

We can see from this table that our Regularized Movie and User Effect Model generates the lowest RMSE, therefore it is our best model. Using the regularized Movie and User Effect Model, we can achieve an RMSE of 0.8648170, which is a significant improvement from our baseline model.

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

method	RMSE
Mean Rating Only	1.0612018
Movie Effect Model	0.9439087
Movie and User Effect Model	0.8653488
Regularized Movie + User Effect Model	0.8648170

Conclusion

Summary

Our goal was to test various machine learning models to find one with the lowest RMSE. The optimal model will be used to predict movie ratings and make a recommendation system. We explored the 10M Movielens dataset to become familiar with the dataset. Then we used r code to visualize different aspects of the data that would be potential predictors for ratings. Based on our findings, we determined that movie and user effects were useful predictors for our model that helped to improve RMSE. Finally, we used regularization in order to account for the effects of movies and users with a low number of ratings. Our final model, with the lowest RMSE of 0.8648170 was the Regularized Movie and User Effect Model.

Limitations and Future Work

It would be useful to explore more potential predictors and models in order to further improve results. We are limited here by the RAM size and processing power of a regular laptop, but this could be explored in the future.

Another possible limitation is the number of variables available in this dataset. With additional variables, we could possibly improve our model. Variables such as lead actors in the movie or user demographics (age and gender) could potentially be useful predictors to include in a new model.