Movie Lens Capstone Project

Eduardo Guiliani

6/17/2021

Overview

The Movie Lens project is part of the HarvardX: PH125.9x Data Science: Capstone course. The aim of the project is to develop and train a recommendation machine learning algorithm to predict a rating given by a user to a set of movies in the data set. The Residual Mean Square Error (RMSE) will be used to evaluate the accuracy of the algorithm. This report will present methods used in exploratory data analysis and visualization, results for the RMSE model and a conclusion based on results of the model. The required criteria for the project is a RMSE < 0.8775, and the optimal criteria is RMSE< 0.86490.

The course provided code that downloaded and cleaned the Movie Lens 10M data set. The code separated the data into two subsets for training (edx) and validation (validation).

Method

Exploratory Data Analysis and Visualization

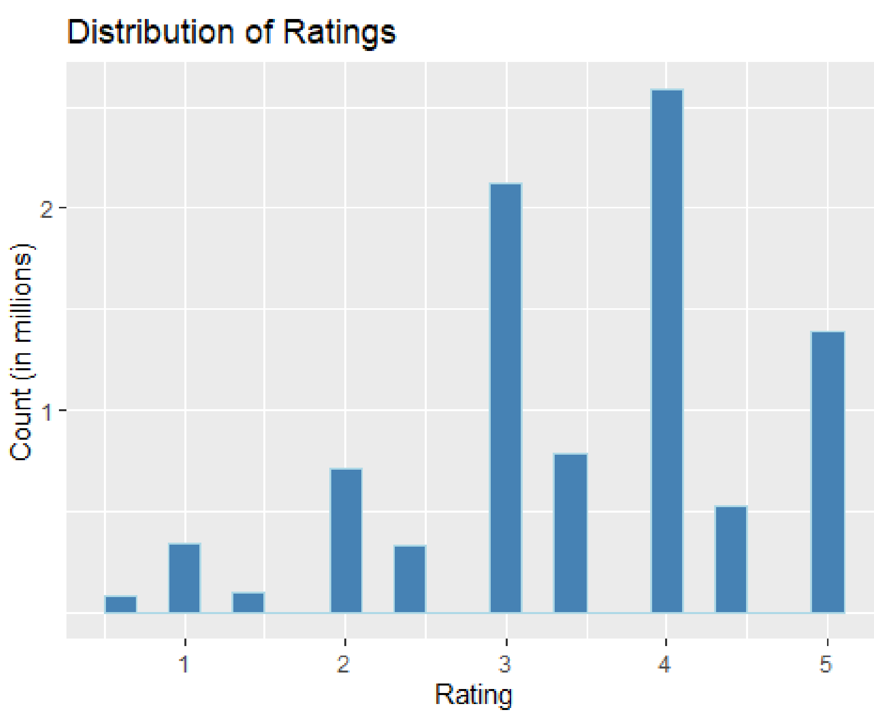
In this section the methods and results of exploratory data analysis performed on the training data set to determine the dimensions and composition of the data set are observed. The method used is presented by the following code

```
# Column summary values for training set
summary(edx)
```

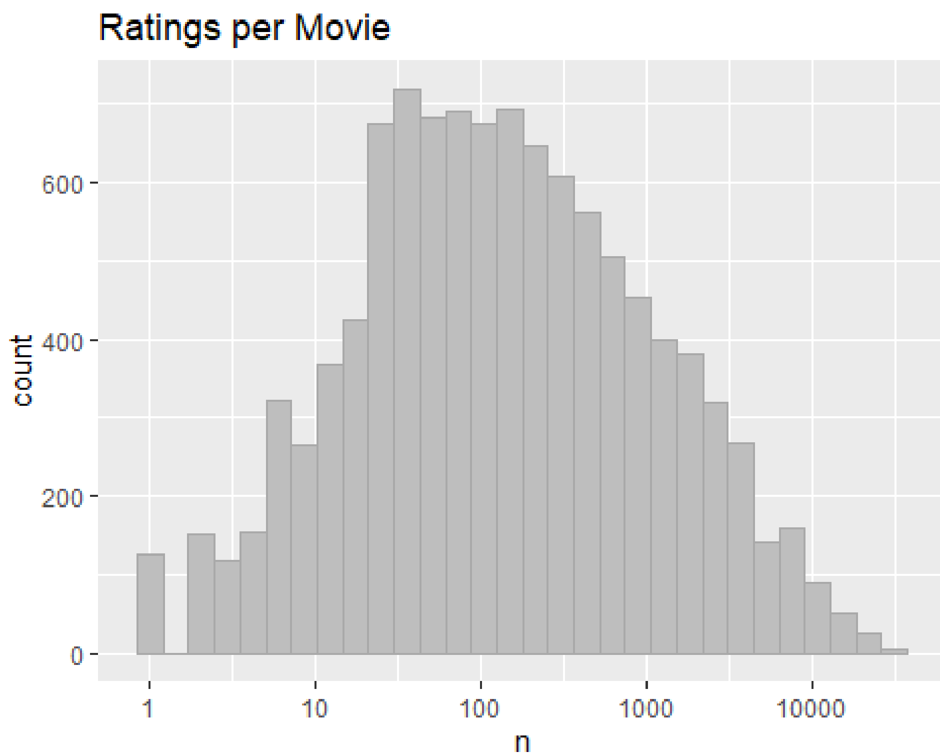
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 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
##
##
##
```

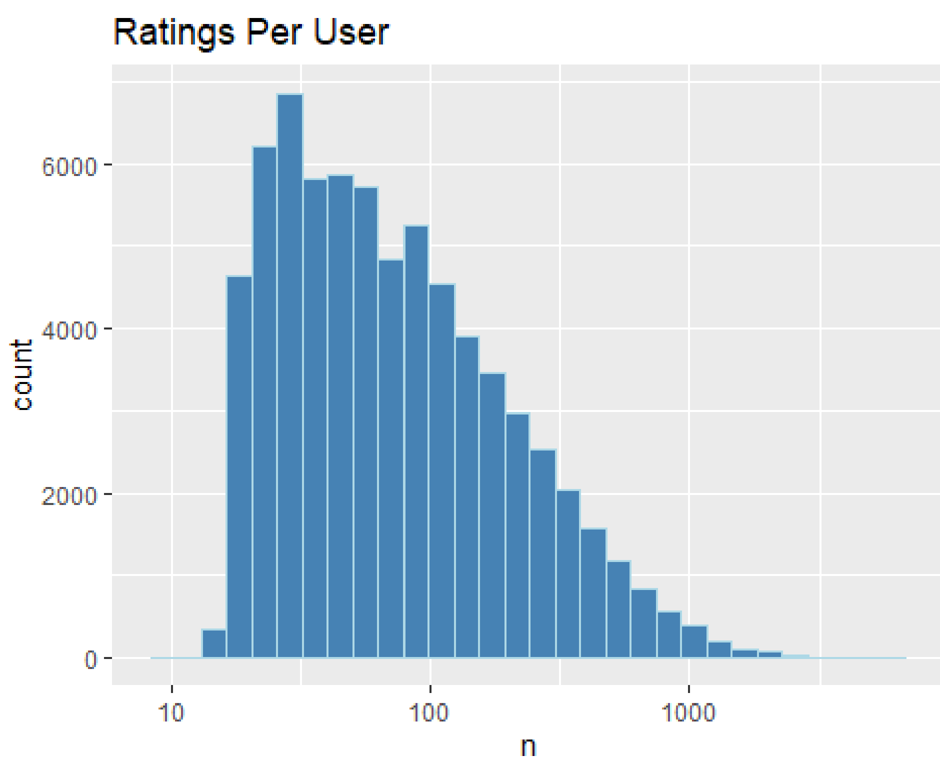
```r
# Dimensions of dataset
cat("The edx dataset has", nrow(edx), "rows and", ncol(edx), "columns.\n")

## The edx dataset has 9000055 rows and 6 columns.

cat("There are", n_distinct(edx$userId), "different users and",
n_distinct(edx$movieId), "different movies in the edx dataset.")

## There are 69878 different users and 10677 different movies in the edx
dataset.

mu <- mean(edx$rating)
cat("The average rating is", mu)

## The average rating is 3.512465
```

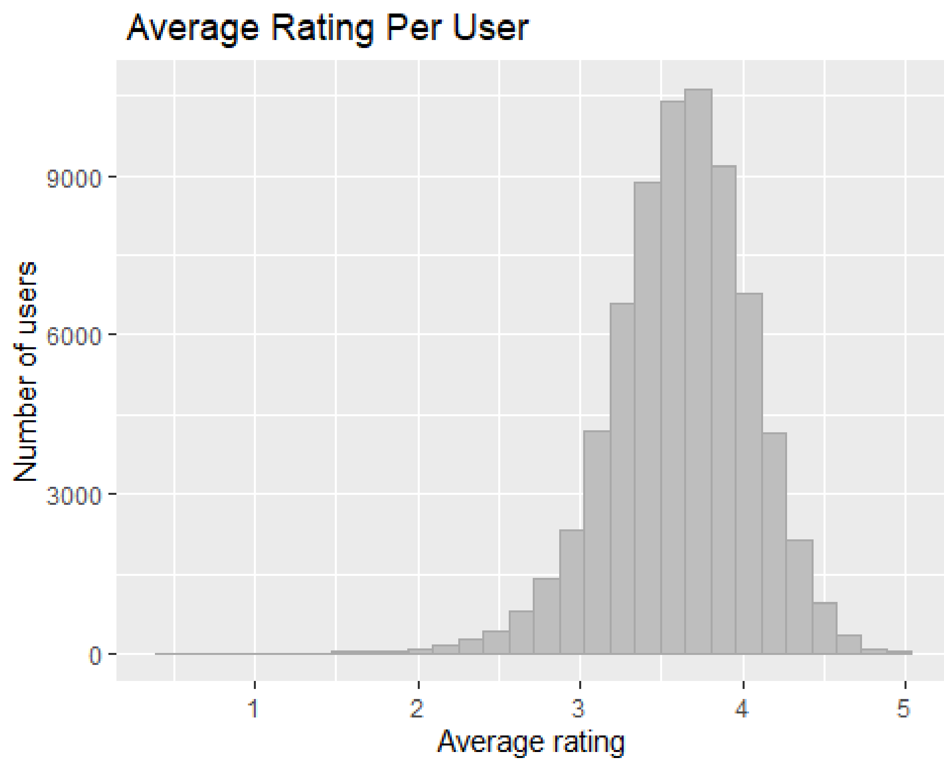
Moreover, a visualization of review ratings was executed to understand the trend in which users gave reviews. From the visualization we can observe that users usually give a full rating rather than a rating and a half, as in they would rather give a 4 or a 5 rather than a 4.5.



Additional data exploration was conducted in order to assess the training data set (edx) based on the amount of user and movie reviews. From the ratings per movie visualization we can observe that there are movies with more ratings than others, thus introducing movie bias based on the number of ratings. Likewise, in the following visualization we can observe there are users that rate more movies than others thus introducing user bias based on the number of ratings. Lastly, from the plot displaying average rating per user we can observe that the average rating is between 3.5 and 4.


Ratings per Movie
count
600
400
200
0
1
10
100
1000
10000
n


Ratings Per User
count
6000
4000
2000
0
10
100
1000
n



To get a better grasp and to further understand the composition of movies per number of reviews and average ratings of these movies, a list was generated of the top 10 reviewed movies. We observe that 9 of the top 10 most reviewed movies are commonly known blockbusters from the 1990's. This table shows that Pulp Fiction is the most reviewed movie, and one may infer from the top 10 that blockbusters are the most rated movies.

```
## # A tibble: 10 x 3
##    title                                                  n
avg
##    <chr>                                              <int>
<dbl>
##  1 Pulp Fiction (1994)                                31362
4.15
##  2 Forrest Gump (1994)                                31079
4.01
##  3 Silence of the Lambs, The (1991)                   30382
4.20
##  4 Jurassic Park (1993)                               29360
3.66
##  5 Shawshank Redemption, The (1994)                   28015
4.46
##  6 Braveheart (1995)                                  26212
4.08
##  7 Fugitive, The (1993)                               25998
4.01
##  8 Terminator 2: Judgment Day (1991)                  25984
```

```
3.93
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
4.22
## 10 Apollo 13 (1995)                                              24284
3.89
```

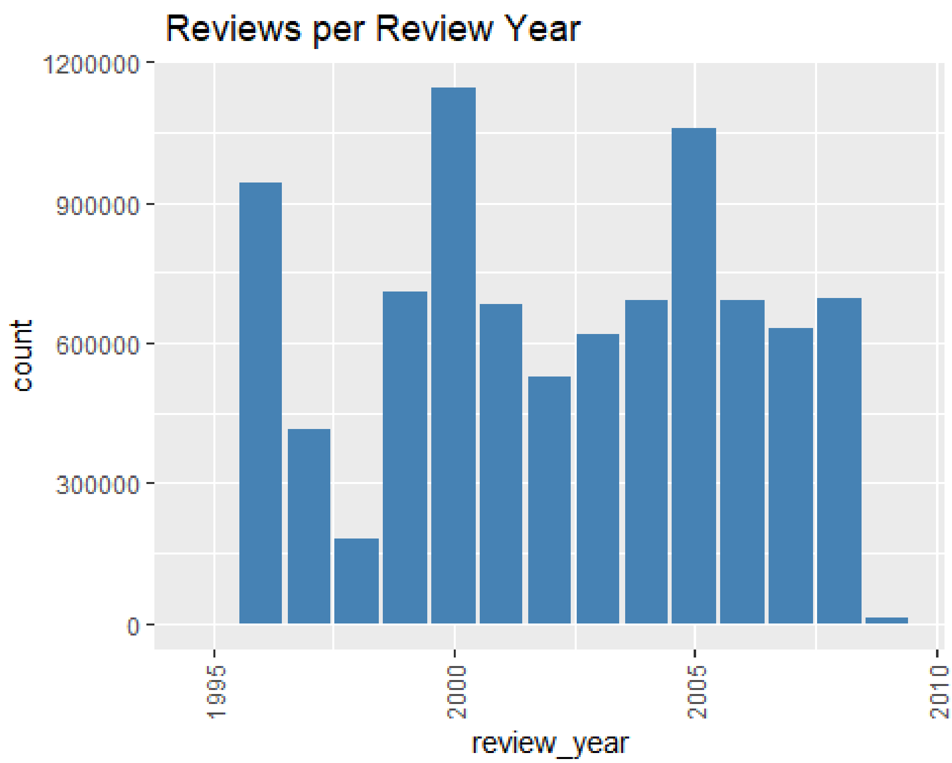
Some additional data wrangling was perfomed to get further insights into the data set. The timestamp was converted to date format and separated into a review year column, and the release year was extracted from the movie title using some regex code and separated into a release year column. The method used is observed in the following code:

```
#Remove timestamp and change to date

  edx_n <- edx %>% mutate(review_year = year(as_datetime(timestamp)))%>%
   select(-timestamp,-genres,-userId)
   head(edx_n)

##    movieId rating                         title review_year
## 1:     122      5              Boomerang (1992)        1996
## 2:     185      5               Net, The (1995)        1996
## 3:     292      5               Outbreak (1995)        1996
## 4:     316      5               Stargate (1994)        1996
## 5:     329      5 Star Trek: Generations (1994)        1996
## 6:     355      5         Flintstones, The (1994)      1996

#Separate year from Movie title

   edx_n1 <- edx_n %>% mutate(release_year =
as.numeric(str_extract(str_extract(title, "[/(]\\d{4}[/)]$"),
regex("\\d{4}"))),title = str_remove(title, "[/(]\\d{4}[/)]$"))
   head(edx_n1)

##    movieId rating                  title review_year release_year
## 1:     122      5              Boomerang        1996         1992
## 2:     185      5               Net, The        1996         1995
## 3:     292      5               Outbreak        1996         1995
## 4:     316      5               Stargate        1996         1994
## 5:     329      5 Star Trek: Generations        1996         1994
## 6:     355      5         Flintstones, The        1996         1994
```

The results show that reviews per year fluctuate and are not incremental per year. Meaning that the number of reviews are not increasing with time.

```
## # A tibble: 15 x 2
##    review_year   count
##          <dbl>   <int>
##  1        2000 1144349
##  2        2005 1059277
##  3        1996  942772
##  4        1999  709893
##  5        2008  696740
```

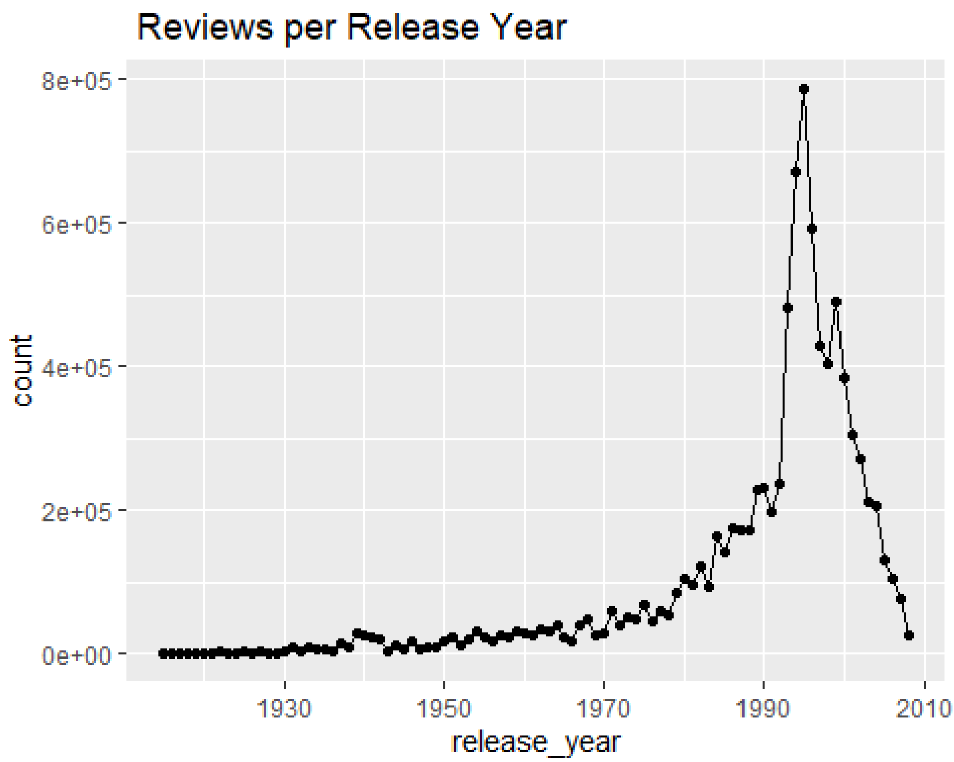
```
## 6 2004 691429
## 7 2006 689315
## 8 2001 683355
## 9 2007 629168
## 10 2003 619938
## 11 2002 524959
## 12 1997 414101
## 13 1998 181634
## 14 2009 13123
## 15 1995 2
```



Furthermore, we can observe that the most reviewed movies were released in the 1990's and have exhibited a mostly downward trend during the 2000's per release year.

```
## Selecting by count

## # A tibble: 10 x 2
##   release_year  count
##          <dbl>  <int>
## 1         1995 786762
## 2         1994 671376
## 3         1996 593518
## 4         1999 489537
## 5         1993 481184
## 6         1997 429751
## 7         1998 402187
## 8         2000 382763
```

```
## 9          2001 305705
## 10         2002 272180
```



The RMSE Model

The evaluation of the predictions were to be executed via an RMSE loss function, defined as:

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

with $\hat{y}_{u,i}$ and $y_{u,i}$ being the predicted and actual ratings, and N, the number of possible combinations between user u and movie i.This function evaluates the square root of the mean of the differences between true and predicted ratings. This equation will define our modeling approach and will serve as the backbone to our improvement of the model.

Results

First Stage

Ratings were approximated by the mean of all ratings in `edx`, which translates to this formula:

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

```
mu <- mean(edx$rating)
mu

## [1] 3.512465

 rmse1 <- RMSE(validation$rating, mu)
 rmse1

## [1] 1.061202
```

Second Stage

We add a parameter to account for the number of ratings for each movie to improve the model. The term is the difference between the average rating and a movie's ratings. The model becomes:

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

where b_i is the new movie effect parameter.

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

     head(movie_avgs)

## # A tibble: 6 x 2
##   movieId    b_i
##     <dbl>  <dbl>
## 1       1  0.415
## 2       2 -0.307
## 3       3 -0.365
## 4       4 -0.648
## 5       5 -0.444
## 6       6  0.303

     # The predicted ratings in y_hat_b_i are based on the mean rating and
movie-dependant parameters b_i

     y_hat_b_i <- validation %>%
     left_join(movie_avgs, by = "movieId") %>%
     mutate(pred = mu + b_i) %>%
     pull(pred)

     rmse2 <- RMSE(validation$rating, y_hat_b_i)
     rmse2

## [1] 0.9439087
```

Third Stage

Given our previous insight that some users review more movies than others, and that the average rating per user varies accross users, we include an additional variable to account for the user effects to further improve the model:

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

where b_u is the user effect parameter.

```
#Takes into account movie effects and user effects by mu + b_i +b_u
# The mean difference of the average rating mu and movie parameter b_i from
each user rating

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

    head(user_avgs)

## # A tibble: 6 x 2
##   userId     b_u
##    <int>   <dbl>
## 1      1  1.68
## 2      2 -0.236
## 3      3  0.264
## 4      4  0.652
## 5      5  0.0853
## 6      6  0.346

# The predicted ratings in y_hat_b_i_b_u are based on the mean rating, movie-
dependant parameters b_i,
#and user dependant parameter b_u

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

    rmse3 <- RMSE(validation$rating, y_hat_b_i_b_u)
    rmse3

## [1] 0.8653488
```

Fourth Stage

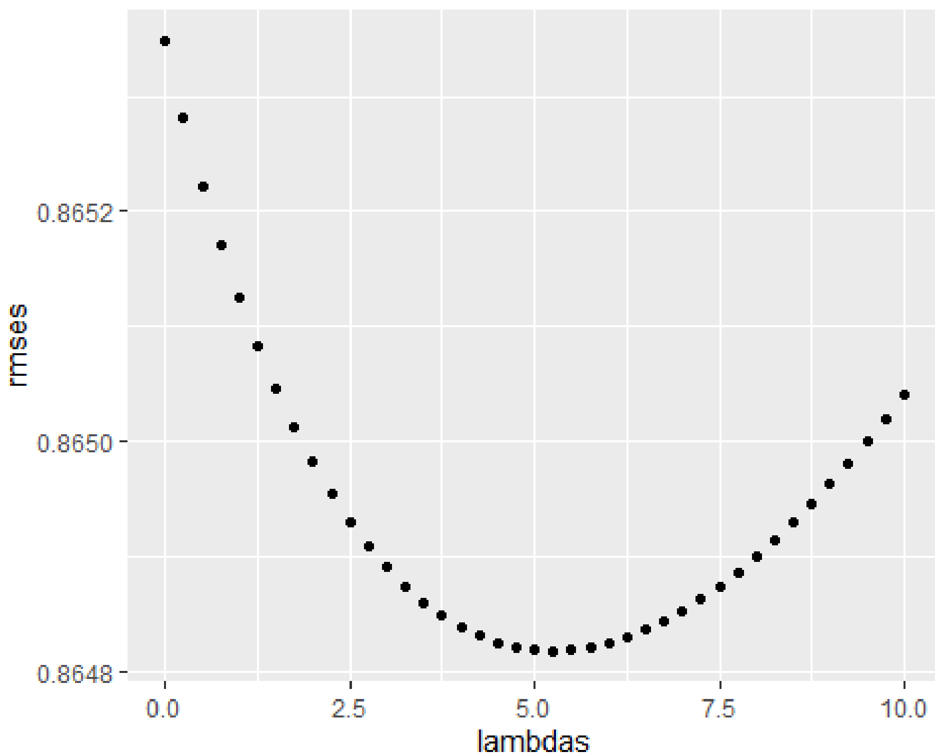
To further improve the model and reach the optimal criteria of a RMSE< 0.86490 we use a regularization method. This method would reduce the effect of less popular movies towards zero. Regularization allows us to penalize large estimates that come from small sample sizes.

$$\hat{b}_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu})$$

```
# We use regularization to take into account the number of ratings per movie
# to diminish the b_i effect of movies with a small number of ratings

    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_regularisation <- min(rmses)
    rmse_regularisation

## [1] 0.864817
```

As we can observe the new model improves the RMSE to achieve the optimal criteria of a RMSE < 0.86490. In order, to visualize the lambda that optimizes the RMSE:

```
qplot(lambdas, rmses)
```




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

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

Conclusion

The objective of the Movie Lens Capstone project was to present a method to minimize an RMSE loss function of the true and predicted ratings of a subset of the MovieLens data set. After data wrangling, exploration and generation of an RMSE function, the model was improved upon to account for movie and user effects, as well as number of ratings to reach the optimal RMSE of < 0.86490 by attaining and RMSE of .864817. Further work could expand upon the system by including additional variables such as genres, release year, and review year.