HarvardX PH125.9x - Data Science: Capstone

Rodrigo Lange

2022-03-01

Contents

1 Introduction

This project is related to the HarvardX Data Science Course PH125.9x. Capstone for this course requires the creation of a movie recommendation system using the 10M version of the MovieLens dataset available at http://grouplens.org/datasets/movielens/10m/.

To train a machine learning algorithm will be used the inputs in training subset (**edx**) to predict movie ratings in the **validation** set.

1.1 Dataset

This is the code to create Train (**edx**) and Final Hold-out (**validation**) Test Sets. I need to develop my algorithm using the **edx** set and predict movie ratings in the **validation** set (the final hold-out test set) as if they were unknown. RMSE will be used to evaluate how close the predictions are to the true values in the **validation** set (the final hold-out test set).

I changed the code so it will check if the data set exists so it will not download again and create a function (my_comma) to display a comma in the thousands place.

```
############################################################
# Create edx set, validation set (final hold-out test 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")

library(tidyverse)
library(caret)
library(data.table)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

# Check if the file exists
datafile <- "MovieLens.RData"
if(!file.exists(datafile))
{
  print("Download")
  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")

  # if using R 4.0 or later:
  movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(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 3.5 or earlier, use `set.seed(1)`
  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)
  save(edx, validation, file = datafile)
} else {
  # If the file exists, just load it
  load(datafile)
}

# Function to display a comma in the thousands place
my_comma <- scales::label_comma(big.mark = ".", decimal.mark = ",")
```

The validation set is 10% of the MovieLens data and the training set is 90%. The number of columns is the same in the **edx** and **validation** datasets.

```
tribble(~"Dataset",~"Rows",~"Columns",
  "training (edx)",     my_comma(nrow(edx)),           my_comma(ncol(edx)),
  "validation",   my_comma(nrow(validation)),     my_comma(ncol(validation))
)
```

```
## # A tibble: 2 x 3
##   Dataset        Rows      Columns
##   <chr>          <chr>     <chr>
## 1 training (edx) 9.000.055 6
## 2 validation     999.999   6
```

The **edx** and **validation** datasets contain 6 columns: “userId”, “movieId”, “rating”, “timestamp”, “title” and “genres”. Each row represents a single rating for a single movie.

```
# Inicial rows - edx dataset
head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
```

```
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                          genres
## 1:               Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

```
# Inicial rows - validation dataset
head(validation)
```

```
##    userId movieId rating timestamp
## 1:      1     231      5 838983392
## 2:      1     480      5 838983653
## 3:      1     586      5 838984068
## 4:      2     151      3 868246450
## 5:      2     858      2 868245645
## 6:      2    1544      3 868245920
##                                                title
## 1:                                    Dumb & Dumber (1994)
## 2:                                    Jurassic Park (1993)
## 3:                                       Home Alone (1990)
## 4:                                          Rob Roy (1995)
## 5:                                    Godfather, The (1972)
## 6: Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
##                                    genres
## 1:                                 Comedy
## 2:        Action|Adventure|Sci-Fi|Thriller
## 3:                          Children|Comedy
## 4:                 Action|Drama|Romance|War
## 5:                             Crime|Drama
## 6: Action|Adventure|Horror|Sci-Fi|Thriller
```

There are no missing values in the **edx** dataset.

```
# Look for NA in the edx dataset
sapply(edx, {function(x) any(is.na(x))})
```

```
##    userId   movieId    rating timestamp     title    genres
##     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE
```

```
# Summarise Data - 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
##
##
##
```

There are no missing values in the **validation** dataset.

```
# Look for NA in the validation dataset
sapply(validation, {function(x) any(is.na(x))})
```

```
##    userId   movieId    rating timestamp     title    genres
##     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE
```

```
# Summarise Data - validation dataset
summary(validation)
```

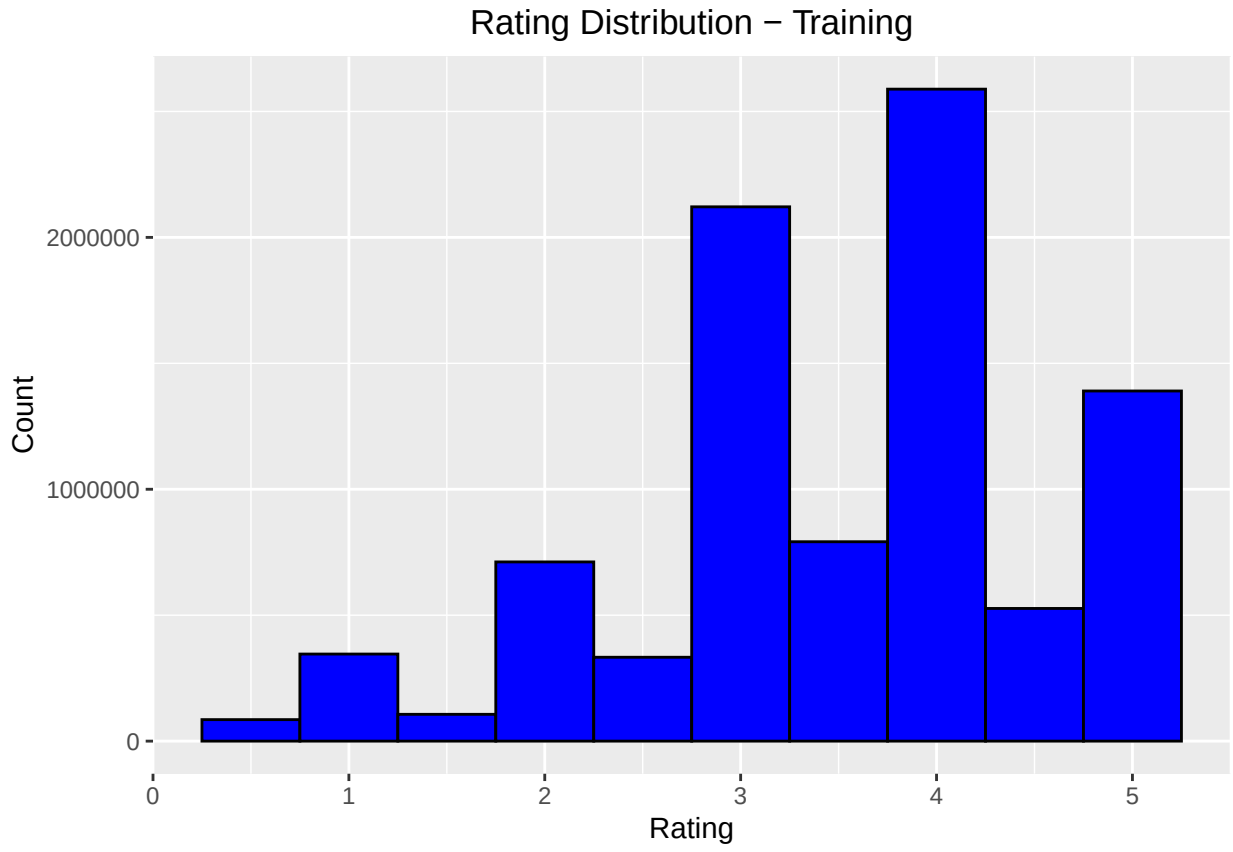
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18096   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.467e+08
##  Median :35768   Median : 1827   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4108   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53621   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres
##  Length:999999      Length:999999
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

2 Analysis

The metric used to evaluate the performance of the algorithm is the Root Mean Square Error or RMSE. The RMSE is a measure of precision, being one of the most used metrics to measure the difference between the values predicted by a model and the values observed. So a smaller RMSE is better than a larger one. Each error is squared in the RMSE. As a result, larger errors have a big effect on the RMSE. The RMSE in this project is expected to be less than 0.86490.

The lowest rating is 0.5 and the highest is 5 in the **edx** dataset. It is more frequent a full star rating than a half star.

```
# Review Training rating distribution
edx %>%
  ggplot(aes(x = rating)) +
  geom_histogram(binwidth=0.5, color="black", fill="blue") +
  scale_y_continuous(labels = function(x) format(x, scientific = FALSE)) +
  theme(plot.title = element_text(hjust = 0.5)) +
  labs(
    title = "Rating Distribution - Training",
    x = "Rating",
    y = "Count"
  )
```

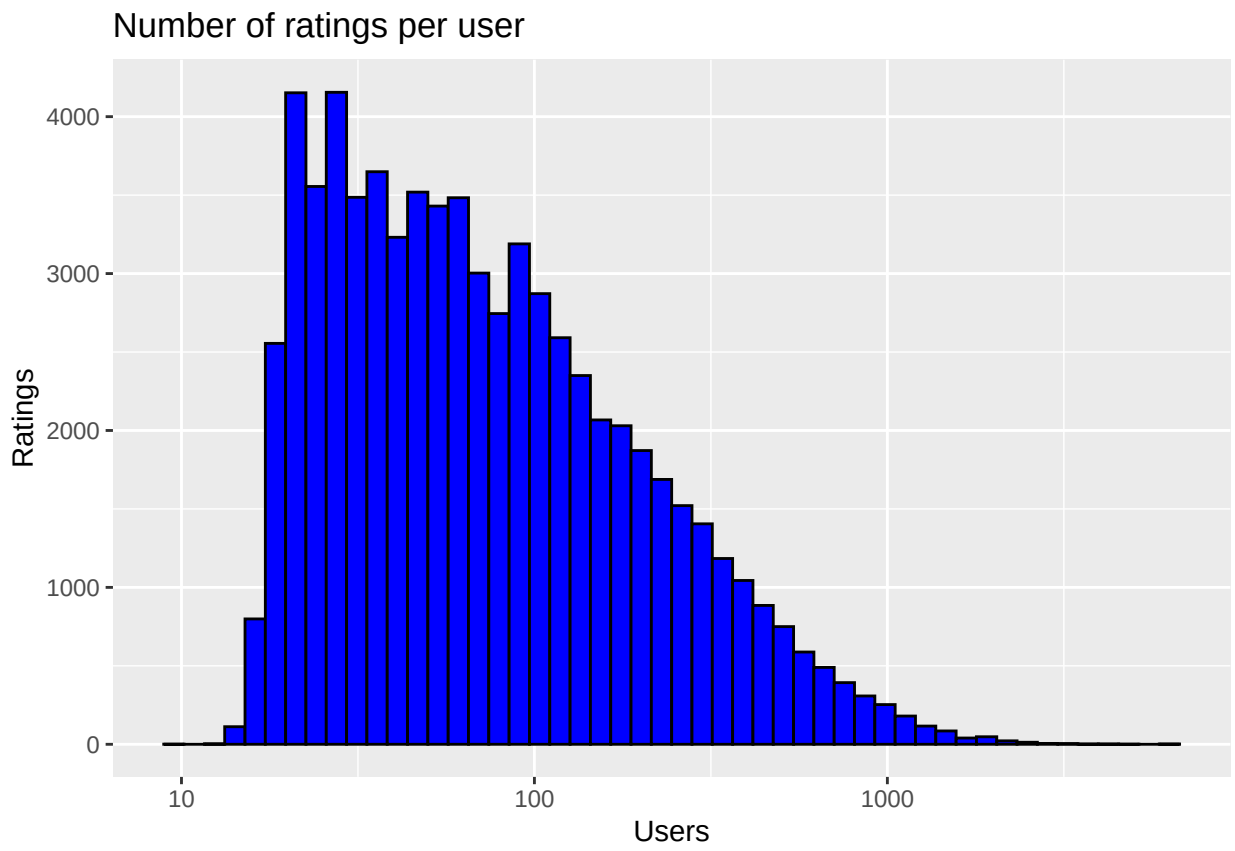


Some users have a very high number of ratings (e.g. more than 5,000 rates). The minimum number of user rates is 10.

```r
# Order the number of ratings per user
edx %>% count(userId) %>% arrange(.,n)
```

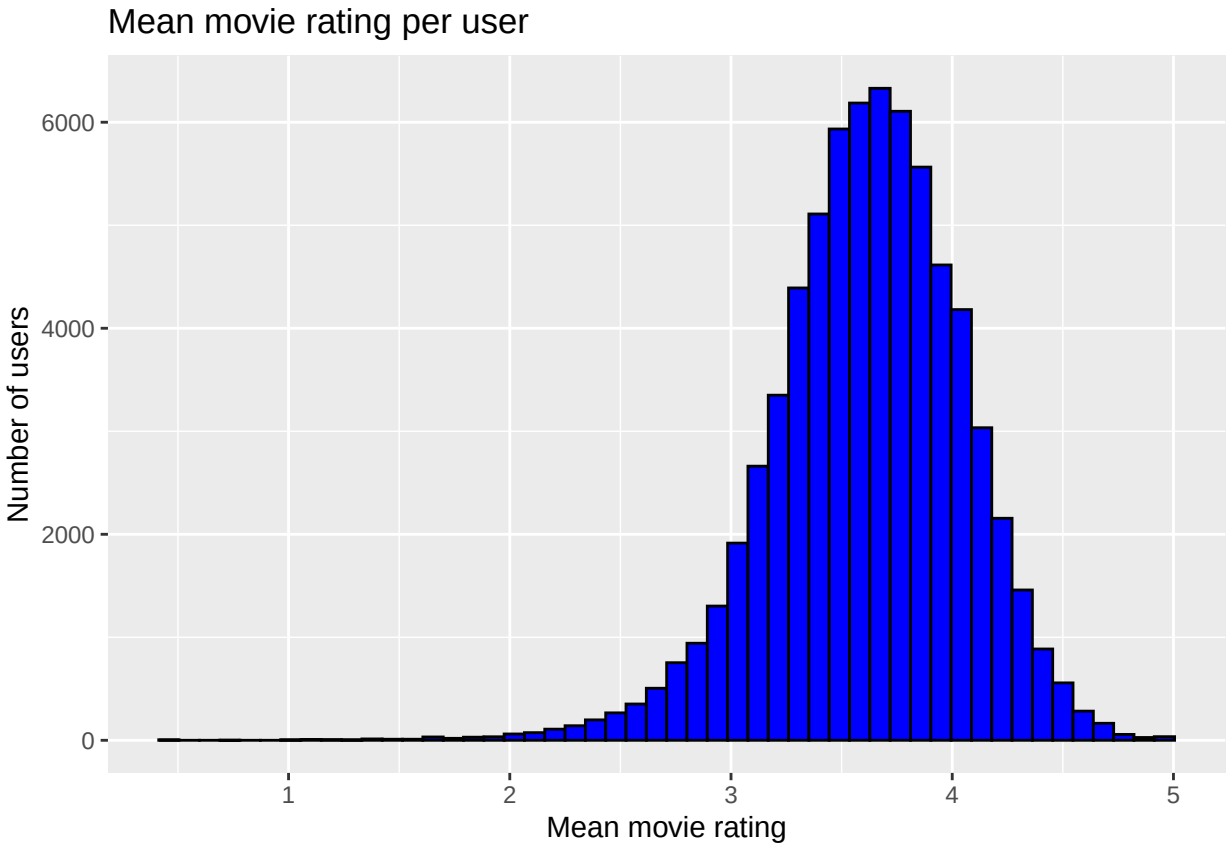
```
##        userId    n
##     1:  62516   10
##     2:  22170   12
##     3:  15719   13
##     4:  50608   13
##     5:    901   14
##    ---
## 69874:  27468 4023
## 69875:  68259 4036
## 69876:  14463 4648
## 69877:  67385 6360
## 69878:  59269 6616
```

```r
# Number of ratings per user
edx %>% count(userId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 50, color = "black", fill="blue") +
scale_x_log10() +
labs(
  title = "Number of ratings per user",
  x = "Users",
  y = "Ratings"
)
```



The users usually gave ratings of 3, 3.5, and 4.

```
edx %>%
group_by(userId) %>%
summarise(mean_rating = mean(rating)) %>%
ggplot(aes(mean_rating)) +
geom_histogram(bins = 50, color = "black", fill = "blue") +
labs(
  title = "Mean movie rating per user",
  x = "Mean movie rating",
  y = "Number of users"
)
```



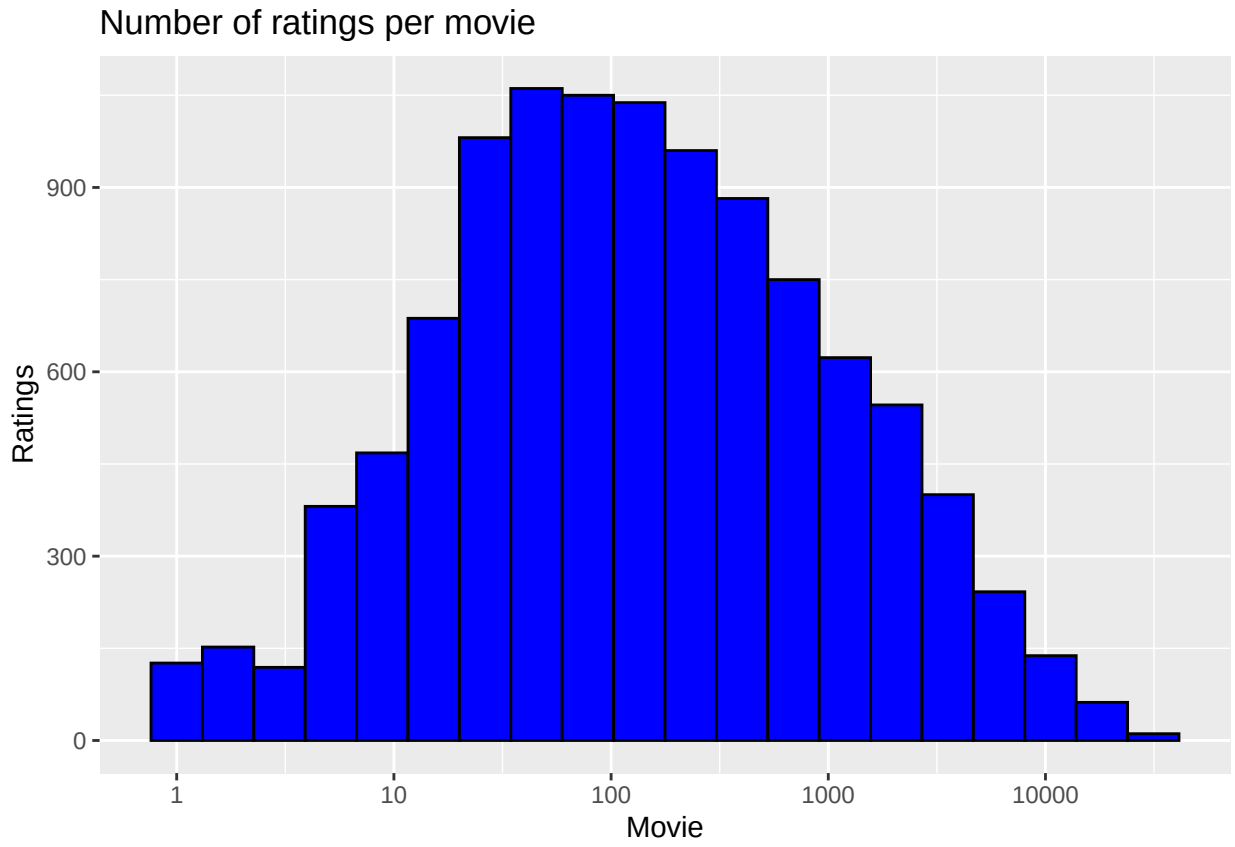
The number of rates per movie varies very much. Some movies have a very low number (e.g. 1) and some movies have more than 30,000 ratings.

```
# Order the number of ratings per user
edx %>% count(movieId) %>% arrange(.,n)
```

```
##        movieId     n
##     1:    3191     1
##     2:    3226     1
##     3:    3234     1
##     4:    3356     1
##     5:    3383     1
##    ---
## 10673:     318 28015
## 10674:     480 29360
## 10675:     593 30382
## 10676:     356 31079
## 10677:     296 31362
```

```
# Number of rates per movie
edx %>%
count(movieId) %>%
ggplot(aes(n)) +
geom_histogram(bins = 20, color = "black", fill="blue") +
scale_x_log10() +
labs(
```

```
  title = "Number of ratings per movie",
  x = "Movie",
  y = "Ratings"
)
```



3 Results

The last section showed that there are movies that are rated more than others, and there are users that rate more than others. The same occurs in the opposite direction. This indicates that there are user and film effects (bias) that can assist in building a good model. I used a function to measure the RMSE:

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

Using the movie and user effect, it is possible to calculate the RMSE in the **edx** dataset.

```
# Model using Movie and User Effect - training (edx) dataset
lambdas <- seq(0, 2, 0.05)
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)
```

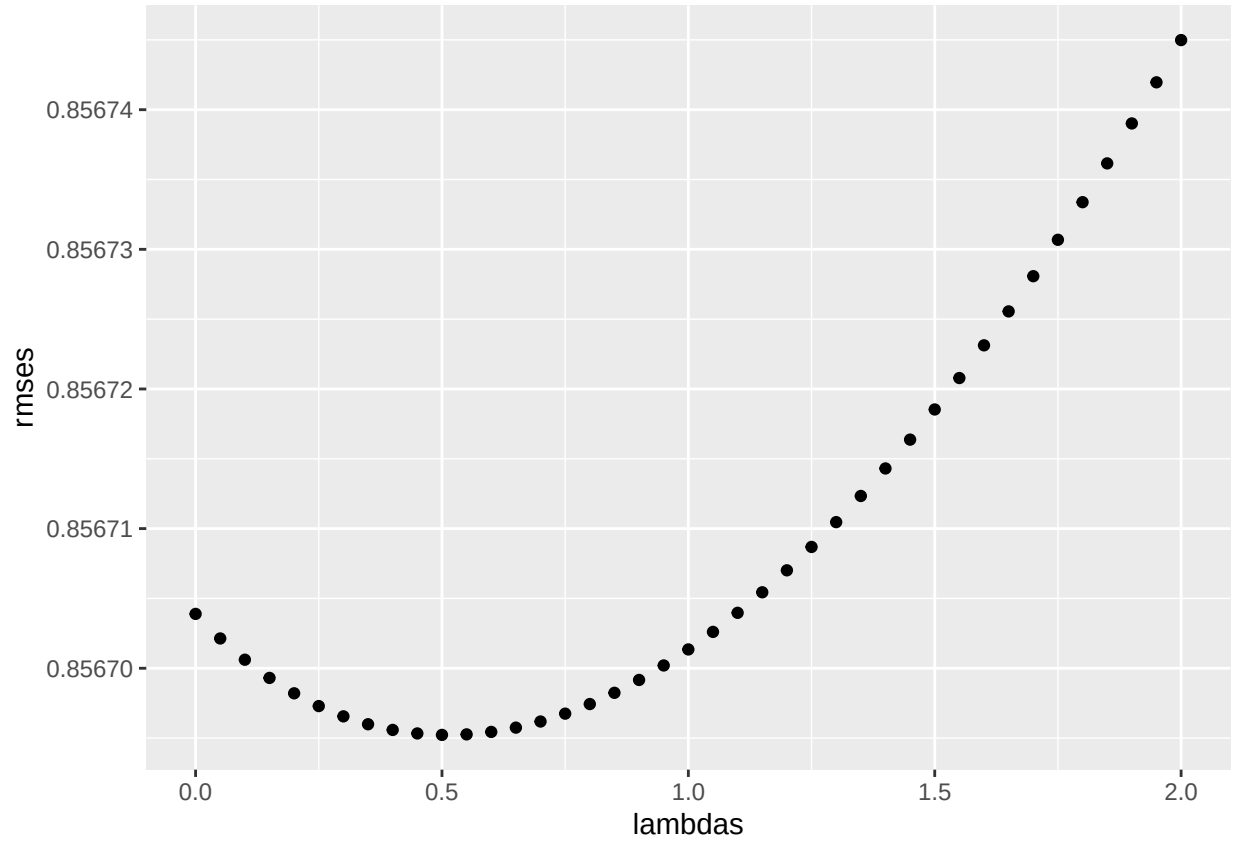
```
  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + l))

  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) %>% .$pred

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

qplot(lambdas, rmses)
```




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

```
## [1] 0.5
```

```
min(rmses)
```

```
## [1] 0.8566952
```

4 Conclusion

Now we can calculate RMSE using the **validation** dataset.

```
# Model using Movie and User Effect - validation dataset
mu <- mean(validation$rating)
l <- 0.15
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) %>% .$pred

RMSE(predicted_ratings, validation$rating)
```

```
## [1] 0.8252108
```

Using the movies and user effect, was possible to get model with a RMSE of 0.8252108.