HarvardX: PH125.9x Data Science: Capstone course
Capstone project MovieLens

Edward Ho

2021/10/15

Introduction

This is the first project of HarvardX: PH125.9x Data Science: Capstone course. We will be using the dataset specified in the assignment to formulate a movie recommendation system. A movie recommendation system is trying to suggest a movie to subscriber for viewing according to user interest. Naturally, a history of user rating on movies and its genres will therefore provide a head start on formulation of the machine learning algorithm.

In this project, we are trying to achieve the goal of formulating a movie recommendation system that has RMSE < 0.86490. The lower the value of RMSE indicates a better fit of the model and its prediction.

The Root Mean Squared Error(RMSE) is defined as following,

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

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

10M version of the MovieLens dataset can be downloaded from http://files.grouplens.org/datasets/movielens/ml-10m.zip. At most of the time, data collected in a real world situation is not complete, having a general understanding of what we data have is a very important steps to save our time in the future processes.

The assignment provided a clean dataset for us to start with using the following attached R code. It will download and clean up the dataset for us. The data is split into training dataset “edx” and testing dataset “validation”.

We will start looking into the data available for us to develop our own ML algorithm.

```
##########################################################
# 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")
```

```r
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(tinytex)) install.packages("tinytex", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)
library(ggplot2)
library(lubridate)
library(tinytex)
# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

dl <- tempfile()
tmp_dir <- tempdir()
dirname(tmp_dir)
#### Read local file , don't need to download and save some time ####
# download.file("file:///C:/Work_proj/workspace_R/capstone_movielens/ml-10m.zip", dl)
# download.file("file:///d:/edward_ho/edx_data_science_cert/capstone/ml-10m.zip", dl)
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 3.6 or earlier:
#movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
#                                            title = as.character(title),
#                                            genres = as.character(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)
```

```r
edx <- rbind(edx, removed)

rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

We will look at the training and testing dataset first. It contains 6 attributes that we can use to develop our machine learning algorithm. Then, we will further exam any NA in the dataset to make sure it is clean and filled.

```r
str(edx)
```

```
## Classes 'data.table' and 'data.frame':   9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ad
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
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
```

```r
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
##
##
##
```

```
anyNA.data.frame(edx)
```

```
## [1] FALSE
```

```
str(validation)
```

```
## Classes 'data.table' and 'data.frame':   999999 obs. of  6 variables:
##  $ userId   : int  1 1 1 2 2 2 3 3 4 4 ...
##  $ movieId  : num  231 480 586 151 858 ...
##  $ rating   : num  5 5 5 3 2 3 3.5 4.5 5 3 ...
##  $ timestamp: int  838983392 838983653 838984068 868246450 868245645 868245920 1136075494 1133571200
##  $ title    : chr  "Dumb & Dumber (1994)" "Jurassic Park (1993)" "Home Alone (1990)" "Rob Roy (1995)
##  $ genres   : chr  "Comedy" "Action|Adventure|Sci-Fi|Thriller" "Children|Comedy" "Action|Drama|Romanc
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
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
```

```
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
##
##
##
```

```r
anyNA.data.frame(validation)
```

```
## [1] FALSE
```

anyNA returns false. It shows that the dataset is clean. However, the genres, timestamp and title columns contain hidden information that may further improve our algorithm. It is the year of release and breakdown of genres. Some minor data massage can be done on these fields.

```r
edx <- edx %>% mutate(rating_time = as.POSIXct(timestamp, origin="1970-01-01"),
                      movie_year2 = str_extract(title, "\\(\\d+\\)")) %>%
  mutate(movie_year=str_extract(movie_year2, "\\d+")) %>% select(-movie_year2)
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         rating_time movie_year
## 1:                Comedy|Romance 1996-08-02 21:24:06       1992
## 2:         Action|Crime|Thriller 1996-08-02 20:58:45       1995
## 3:  Action|Drama|Sci-Fi|Thriller 1996-08-02 20:57:01       1995
## 4:       Action|Adventure|Sci-Fi 1996-08-02 20:56:32       1994
## 5: Action|Adventure|Drama|Sci-Fi 1996-08-02 20:56:32       1994
## 6:       Children|Comedy|Fantasy 1996-08-02 21:14:34       1994
```

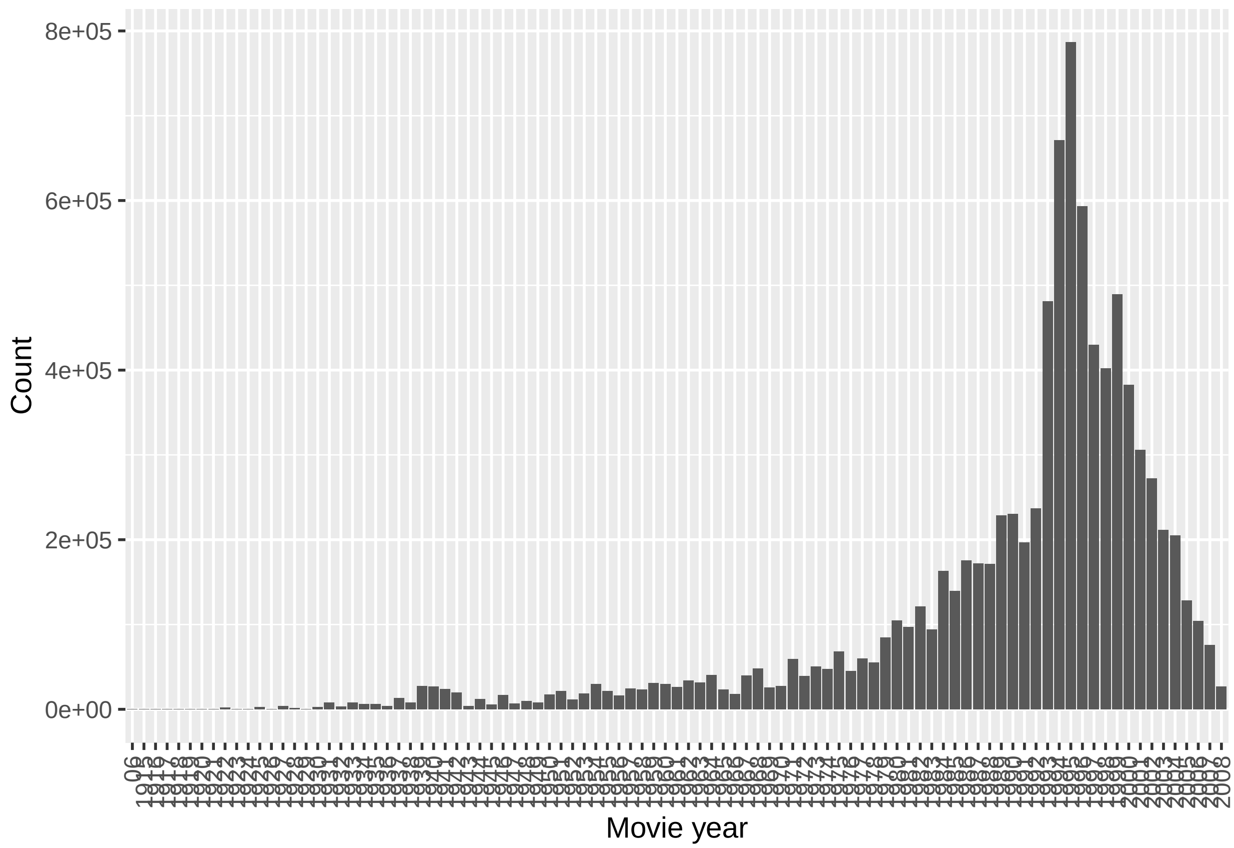
```r
validation <- validation %>% mutate(rating_time = as.POSIXct(timestamp, origin="1970-01-01"),
                                    movie_year2 = str_extract(title, "\\(\\d+\\)")) %>%
  mutate(movie_year=str_extract(movie_year2, "\\d+")) %>% select(-movie_year2)
head(validation)
```

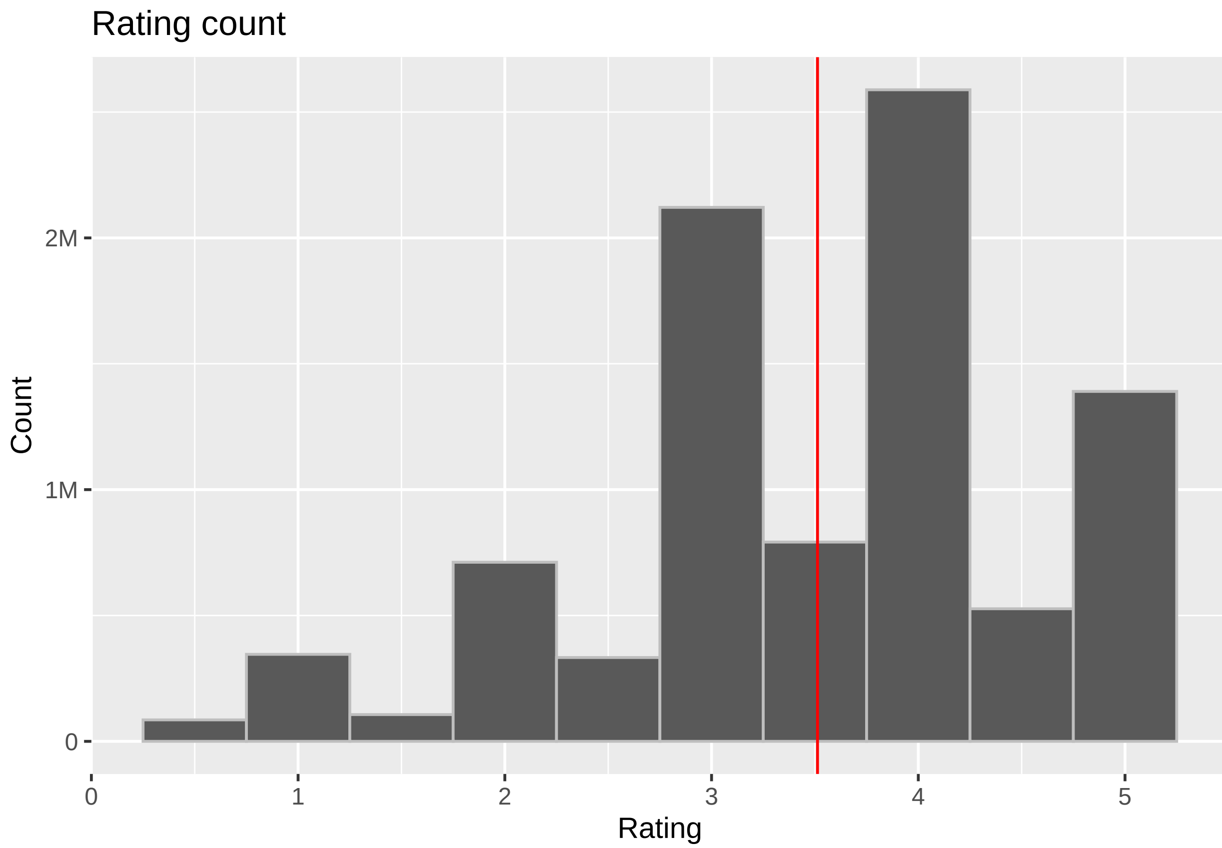
```
##    userId movieId rating timestamp
## 1:      1     231      5 838983392
## 2:      1     480      5 838983653
## 3:      1     586      5 838984068
## 4:      2     151      3 868246450
## 5:      2     858      2 868245645
## 6:      2    1544      3 868245920
##                                                   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         rating_time movie_year
## 1:                                   Comedy 1996-08-02 20:56:32       1994
## 2:        Action|Adventure|Sci-Fi|Thriller 1996-08-02 21:00:53       1993
## 3:                          Children|Comedy 1996-08-02 21:07:48       1990
## 4:               Action|Drama|Romance|War 1997-07-07 13:34:10       1995
## 5:                              Crime|Drama 1997-07-07 13:20:45       1972
## 6: Action|Adventure|Horror|Sci-Fi|Thriller 1997-07-07 13:25:20       1997
```

Visualization of data

"Picture worth a thousand words". Visualization is a very important step to gain understanding of our data. We use ggplot2 to plot graphs of the edx dataset.





The mean value of rating is about 3.5, and it is shown as a red line in the historgram above.

Methods and analysis

First method we will adopt as our algorithm for a movie recommendation system is the average of the movie rating. We just simply take the mean of the rating column in our edx dataset.

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

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

RMSE(validation$rating, mu)
```

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

```
method_average <- RMSE(validation$rating, mu)
rmse_results1 <- data.frame(method = "The average rating", RMSE = method_average)
rmse_results1
```

```
##               method     RMSE
## 1 The average rating 1.061202
```

The RMSE is 1.0612. This value is very far away from our target value.

Second method we will try to use is average + movie effect.

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

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

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

rmse_results2 <- data.frame(method = "Movie effects", RMSE = method_movie_effects)
rmse_results2
```

```
##          method      RMSE
## 1 Movie effects 0.9439087
```

The RMSE is 0.9439. This value is still very far away from our target value. In order to improve our RMSE. We are going to add the third attribute from the edx data to improve our algorithm.

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

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

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

```
rmse_results3 <- data.frame(method = "Movie_user_effects", RMSE = method_movie_user_effects)
```

The third method generated RMSE = 0.8653488. We still have a room for improvement. Therefore, we try to throw in additional attributes and hoping it will generate a RMSE below 0.865.

The forth attribute we use is movie released year which is hidding inside the title column in the original dataset.

```
## Try to add other factor: movie effects and user effects and year effects
movie_year_avgs <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(movie_year) %>%
  summarize(b_y = mean(rating - mu - b_i - b_u))
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(movie_year_avgs, by='movie_year') %>%
  mutate(pred = mu + b_i + b_u + b_y) %>%
```

```
  pull(pred)
method_movie_user_year_effects <- RMSE(validation$rating, predicted_ratings)
method_movie_user_year_effects
```

```
## [1] 0.8650034
```

```
rmse_results4 <- data.frame(method = "Movie_user_year_effects",
                            RMSE = method_movie_user_year_effects)
```

The improvement of employing movie released year attribute does not change RMSE much.

```
improvement <- (method_movie_user_year_effects / method_movie_user_effects)
```

We can see that there is less than 1 percent improvement. Adding more attributes into the lm model is approaching its limit. We cannot shrink the value of RMSE further more with additional attributes in the lm model. We need to another techniques to improve our algorithm.

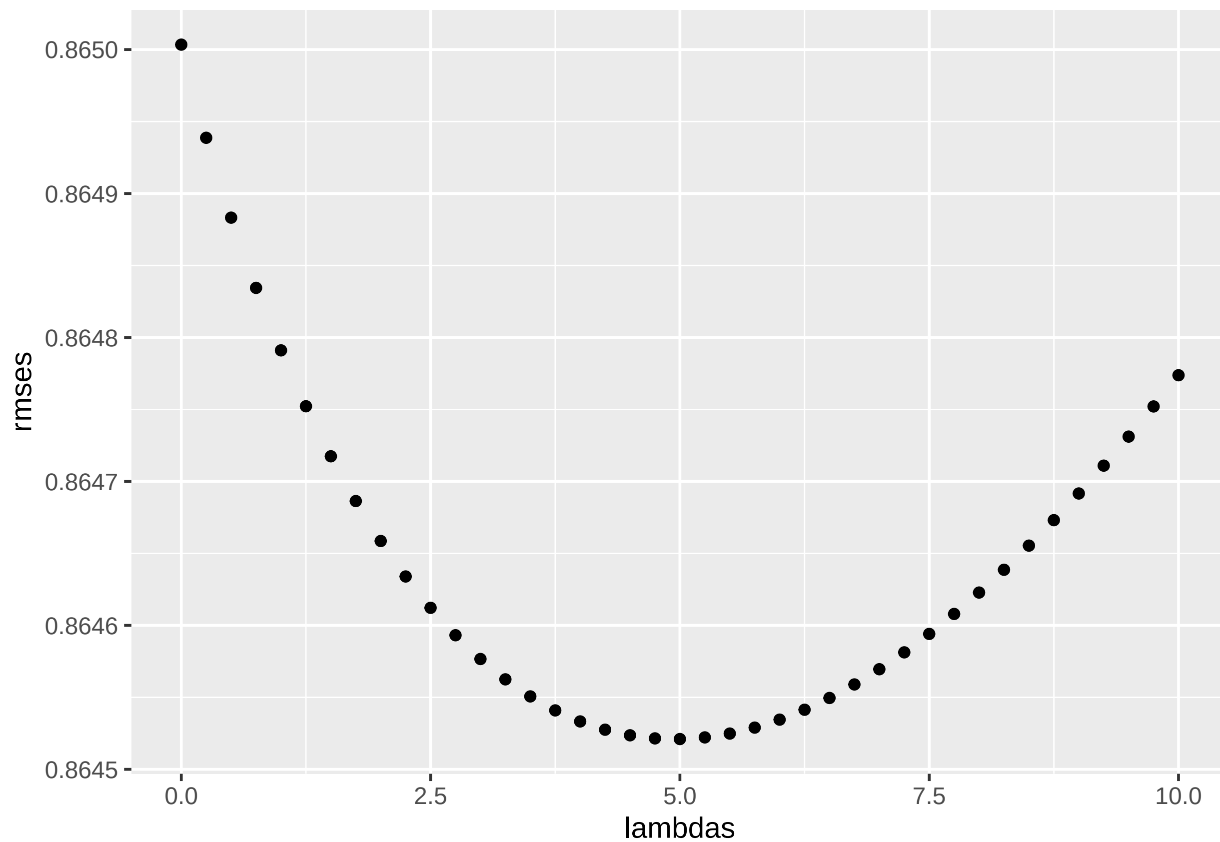
Regularization

A technique called Regularization allows us to penalize effects from small sample size. In our existing dataset the effects from small sample size distort our lm algorithm. We need to adjust, or regularize the distortion. In this section, we use the regularization technique to try to improve the RMSE value. It shrinks the effects of some parameters, in our case, when a movie is only rated by small number of users, by adding a penalty to this effects. It is possible to reduce it with proper value of lambda.

```
## assigns a sequence of number into lambdas variable
lambdas <- seq(0, 10, 0.25)

## passing in lambdas and generate a vector of RMSEs, then we will use this vector to plot
##  our lambdas graph.
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)
# movie effect
  b_i <- edx %>% group_by(movieId) %>% summarize(b_i = sum(rating - mu)/(n() + l))
# user effect
  b_u <- edx %>% left_join(b_i, by="movieId") %>% group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n() + l))
# movie released year effect
  b_y <- edx %>% left_join(b_i, by="movieId") %>% left_join(b_u, by="userId") %>%
    group_by(movie_year) %>% summarize(b_y = sum(rating - b_i - b_u - mu)/(n() + l))

  predicted_ratings <- validation %>% left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>% left_join(b_y, by='movie_year') %>%
    mutate(pred = mu + b_i + b_u + b_y) %>% pull(pred)

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

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

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

```r
b_u <- edx %>% left_join(b_i, by="movieId") %>% group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu)/(n() + lambda))

b_y <- edx %>% left_join(b_i, by="movieId") %>% left_join(b_u, by="userId") %>%
  group_by(movie_year) %>% summarize(b_y = sum(rating - b_i - b_u - mu)/(n() + lambda))

predicted_ratings <- validation %>% left_join(b_i, by='movieId') %>%
  left_join(b_u, by='userId') %>% left_join(b_y, by='movie_year') %>%
  mutate(pred = mu + b_i + b_u + b_y) %>% pull(pred)

method_reg_user_movie_year <- RMSE(validation$rating, predicted_ratings)

rmse_results5 <- data.frame(method = "Movie_reg_user_year_effects", RMSE = method_reg_user_movie_year)
```

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



As shown in the graph of the variable lambda, the lowest is at 5.

We will apply lambda <- 5 to our final algorithm.

Result

It shows improvement when we add more attribute into the lm algorithms. However, it stops improving when the 3rd attribute added. We are forced to review our algorithm and we need to apply regularization technique to improve the RMSE. The final result of the RMSE is 0.8645625.

```r
rmse_all_results = rbind(rmse_results1, rmse_results2, rmse_results3,
                         rmse_results3, rmse_results4, rmse_results5)
rmse_all_results
```

```
##                         method      RMSE
## 1            Just the average 1.0612018
## 2               Movie effects 0.9439087
## 3           Movie_user_effects 0.8653488
## 4           Movie_user_effects 0.8653488
## 5      Movie_user_year_effects 0.8650034
## 6 Movie_reg_user_year_effects 0.8645210
```

Conclusion

The target of this project is to develop an algorithm that can achieve RMSE as small as possible. Through the use of linear model with multiple attributes, we can arrive RMSE 0.8645625.