MovieLens Project

Hieu Nguyen

10/11/2021

Contents

Introduction

The aim of this project is to create movie recommendation using the MovieLens dataset. We will analyze the past interaction between users and movies to build recommendation system.

Goals

Root Mean Square Error (RMSE) will be used to as key metric to measure performance of different algorithms. Our objective is to analyze edx dataset and find best algorithm with lowest RMSE and then apply it to validation dataset (the final hold-out test set) to predict rating of each movie by each user. RMSE will also be used to evaluate how close the predictions with actual rating in validation dataset. Our target RMSE is smaller than 0.86490.

Dataset

We will use MovieLens 10M Dataset from this this site: https://grouplens.org/datasets/movielens/10m/

Dataset information:

- Number of instances: 10,000,054
- Number of attributes: 6
- Attribute information:
  1. userId: ID of each user that given rating to each movie
  2. movieId: ID of each movie
  3. rating: rating that user give to each movie
  4. timestamp: time that user give rating
  5. title: title of movie with year of production information
  6. genres: genres of movie

Key steps

The key steps are the following:

1. Installation of required packages and loading of libraries
2. Downloading and formatting the **movilens** dataset for further processing
3. Analyze to understand the datasets and get insights of its features
4. Partition **movielens** dataset into **edx** (90%) and **validation** (10%)
5. Partition **edx** dataset into **train_set** (90%) and **test_set** (10%)

6. Applying different algorithms on **train_set** dataset and using RMSE to evaluate on **test_set** dataset to find best algorithm that give smallest RMSE

7. Applying best algorithm on **edx** dataset and evaluate how close the predictions with actual rating in **validation** dataset

Movielens dataset analysis

Installing and loading required libraries

Loading required package libraries to use its functions in our Movielens data analysis

```
# Check and install packages if not exist
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")
if(!require(lubridate)) install.packages("lubridate",
                                          repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(lubridate)
```

Movielens Dataset download and Preparation

Movielens dataset can be downloaded from this site 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(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))

movielens <- left_join(ratings, movies, by = "movieId")
```

Data exploration and visualisation

We will extract information from current data and add back to dataset to check relationship of these information with rating:
- year: the year that movie was made: extracted from title
- time that user give rating: extracted from timestamp
- week: week number: 1 - 52
- wday: Monday, Tuesday … Sunday in form week-day number 1, ..7
- hour: 0, 1 … 23

```r
movielens <- movielens %>%
  mutate(year = strtoi(substr(title, nchar(title) - 4, nchar(title) -1)),
         week = week(as_datetime(timestamp)),
         wday = wday(as_datetime(timestamp)),
         hour = hour(as_datetime(timestamp)))
```

Structure of movielens dataset

```r
str(movielens)
```

```
## Classes 'data.table' and 'data.frame':   10000054 obs. of  10 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 231 292 316 329 355 356 362 364 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983392 838983421 838983392 838983392 838984474 838983653 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb & Dumber (1994)" "Outbreak (1995)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Comedy" "Action|Drama|Sci-Fi|Thriller"
##  $ year     : int  1992 1995 1994 1995 1994 1994 1994 1994 1994 1994 ...
##  $ week     : num  31 31 31 31 31 31 31 31 31 31 ...
##  $ wday     : num  6 6 6 6 6 6 6 6 6 6 ...
##  $ hour     : int  11 10 10 10 10 10 11 11 11 11 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
movielens %>% as_tibble()
```

```
## # A tibble: 10,000,054 x 10
##    userId movieId rating timestamp title     genres       year  week  wday  hour
##     <int>   <dbl>  <dbl>     <int> <chr>     <chr>       <int> <dbl> <dbl> <int>
##  1      1     122      5 838985046 Boomeran~ Comedy|Rom~  1992    31     6    11
##  2      1     185      5 838983525 Net, The~ Action|Cri~  1995    31     6    10
##  3      1     231      5 838983392 Dumb & D~ Comedy       1994    31     6    10
##  4      1     292      5 838983421 Outbreak~ Action|Dra~  1995    31     6    10
##  5      1     316      5 838983392 Stargate~ Action|Adv~  1994    31     6    10
##  6      1     329      5 838983392 Star Tre~ Action|Adv~  1994    31     6    10
##  7      1     355      5 838984474 Flintsto~ Children|C~  1994    31     6    11
##  8      1     356      5 838983653 Forrest ~ Comedy|Dra~  1994    31     6    11
##  9      1     362      5 838984885 Jungle B~ Adventure|~  1994    31     6    11
## 10      1     364      5 838983707 Lion Kin~ Adventure|~  1994    31     6    11
## # ... with 10,000,044 more rows
```

Number of unique userId, movieId and genres in movielens dataset

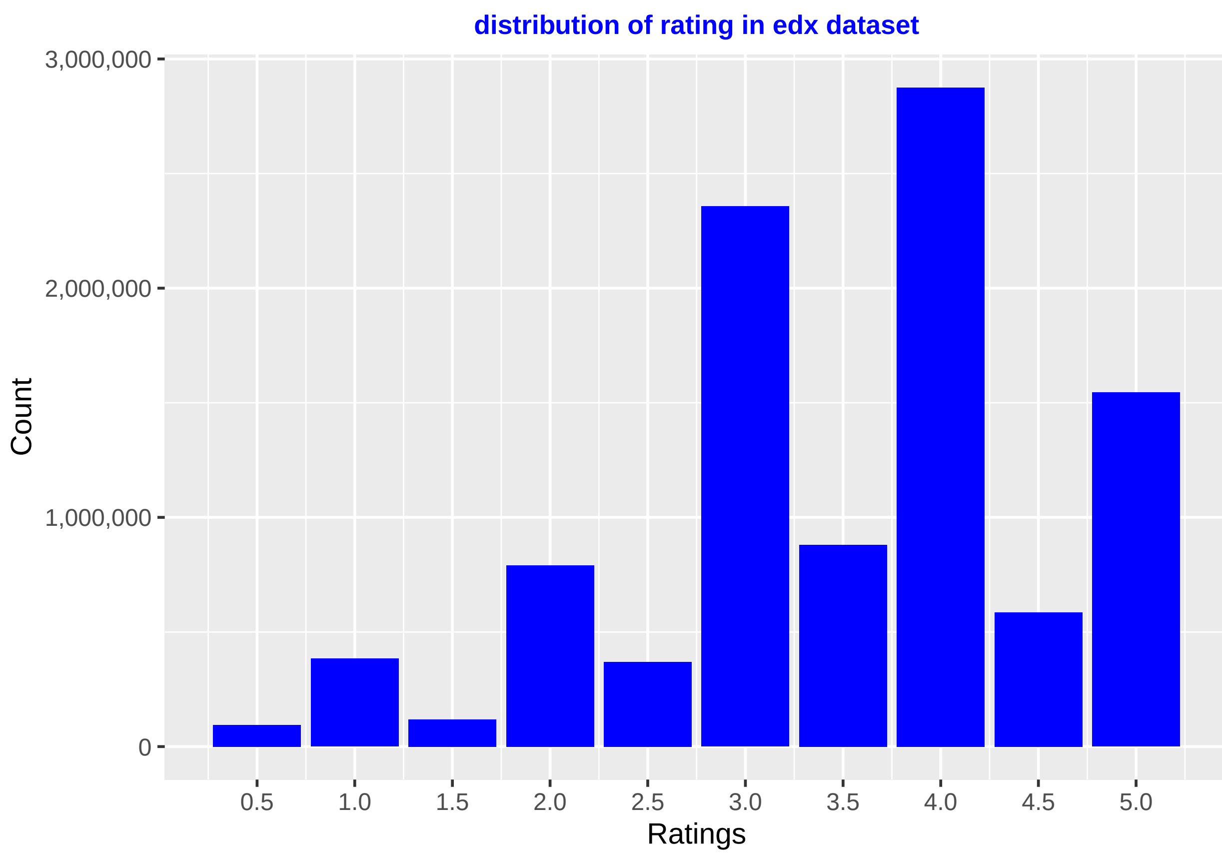
```r
data.frame(userId = n_distinct(movielens$userId),
           movieId = n_distinct(movielens$movieId),
           genres = n_distinct(movielens$genres),
           row.names = "MovieLens: Number of Unique") %>% knitr::kable()
```

	userId	movieId	genres
MovieLens: Number of Unique	69878	10677	797

Distribution of rating in movielens dataset

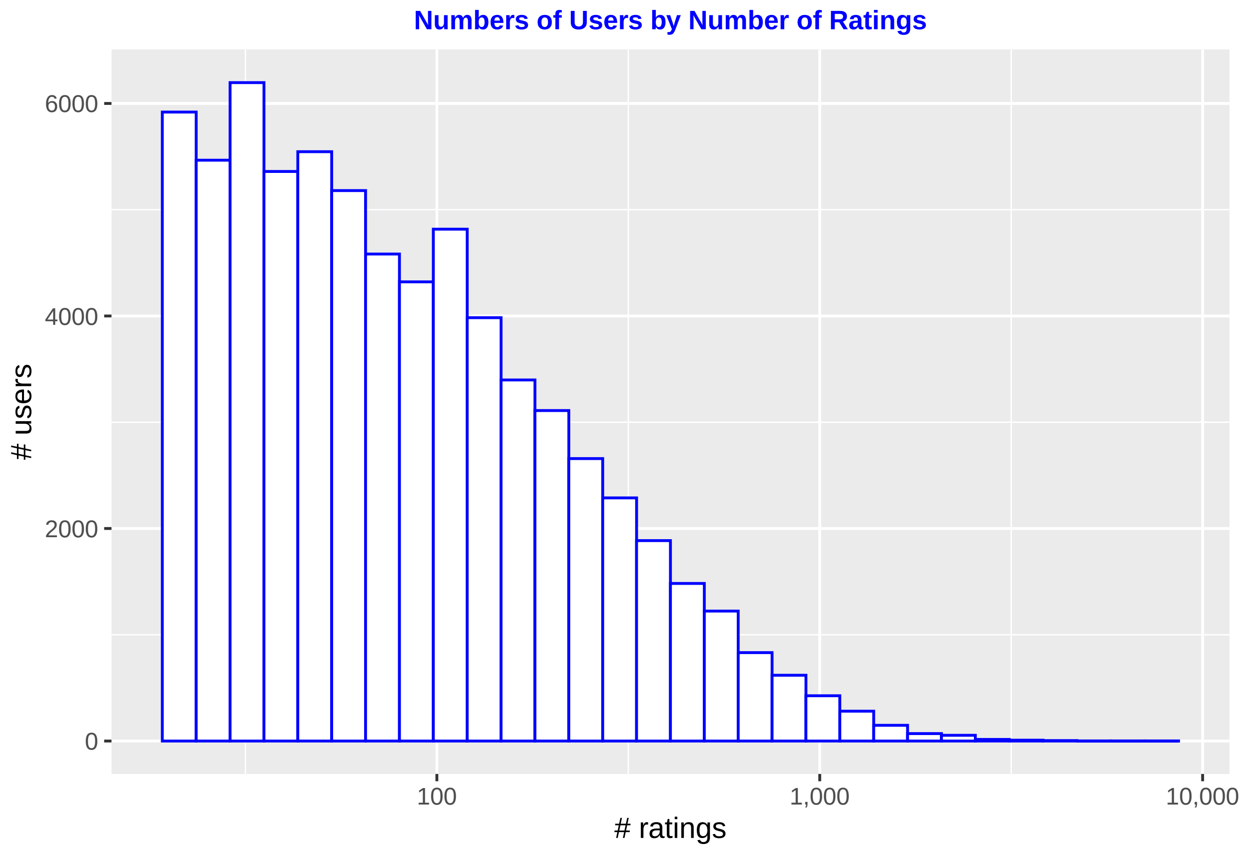
We observe that most of ratings are given from 3 to 5 and highest distribution is rating at 4 stars.

```
movielens %>% group_by(rating) %>% summarize(count = n()) %>%
  ggplot(aes(rating, count)) + geom_col(fill = "blue") +
  xlab("Ratings") + ylab("Count") + ggtitle("distribution of rating in edx dataset") +
  scale_x_continuous(breaks = seq(0,5,0.5)) +
  scale_y_continuous(labels = scales::comma) +
  theme(plot.title = element_text(hjust = 0.5, size = 10, color = "blue", face = "bold"))
```



Plotting number of Users by Number of Ratings

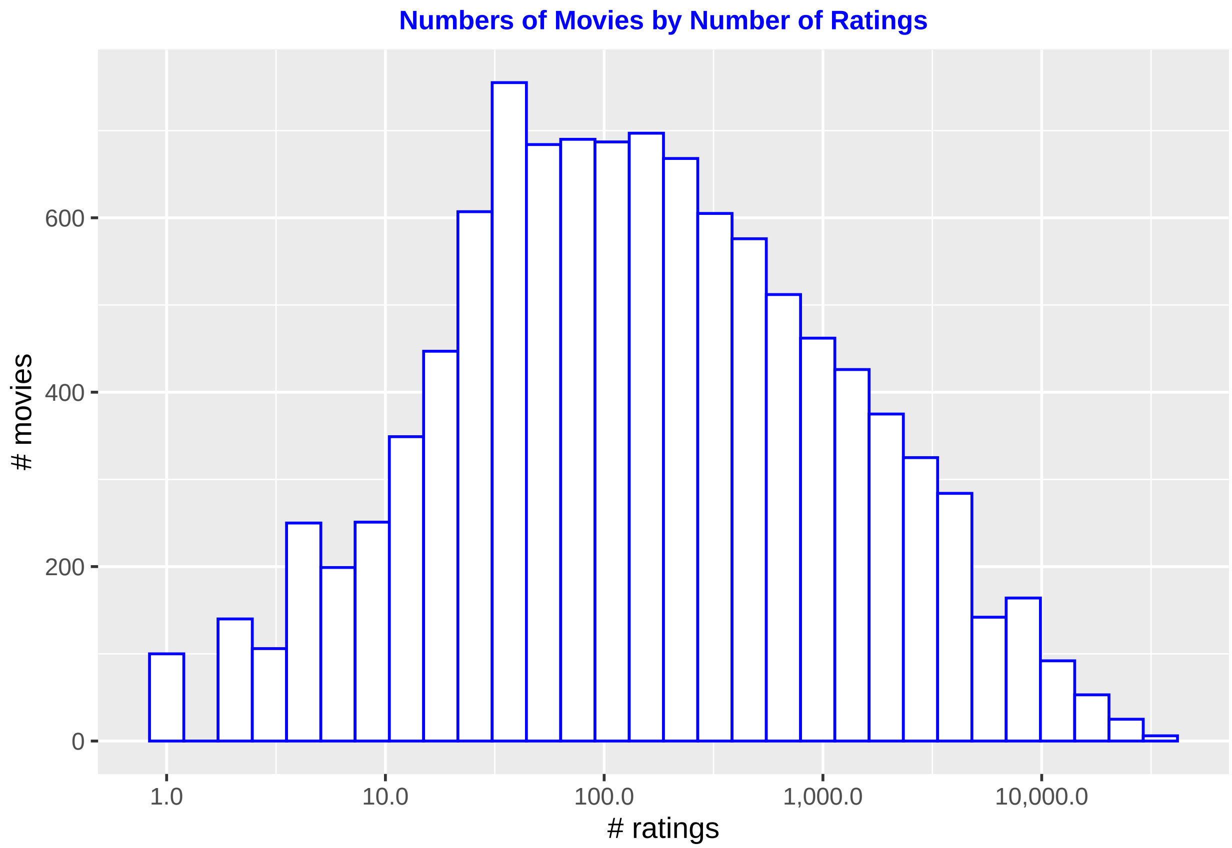
```
movielens %>% group_by(userId) %>% summarize(count = n()) %>%
  ggplot(aes(count)) + geom_histogram(color = "blue", fill="white") +
  xlab("# ratings") + ylab("# users") + ggtitle("Numbers of Users by Number of Ratings") +
  theme(plot.title = element_text(hjust = 0.5, size = 10, color = "blue", face = "bold")) +
  scale_x_log10(labels = scales::comma)
```



Plot Numbers of Movies by Number of Ratings

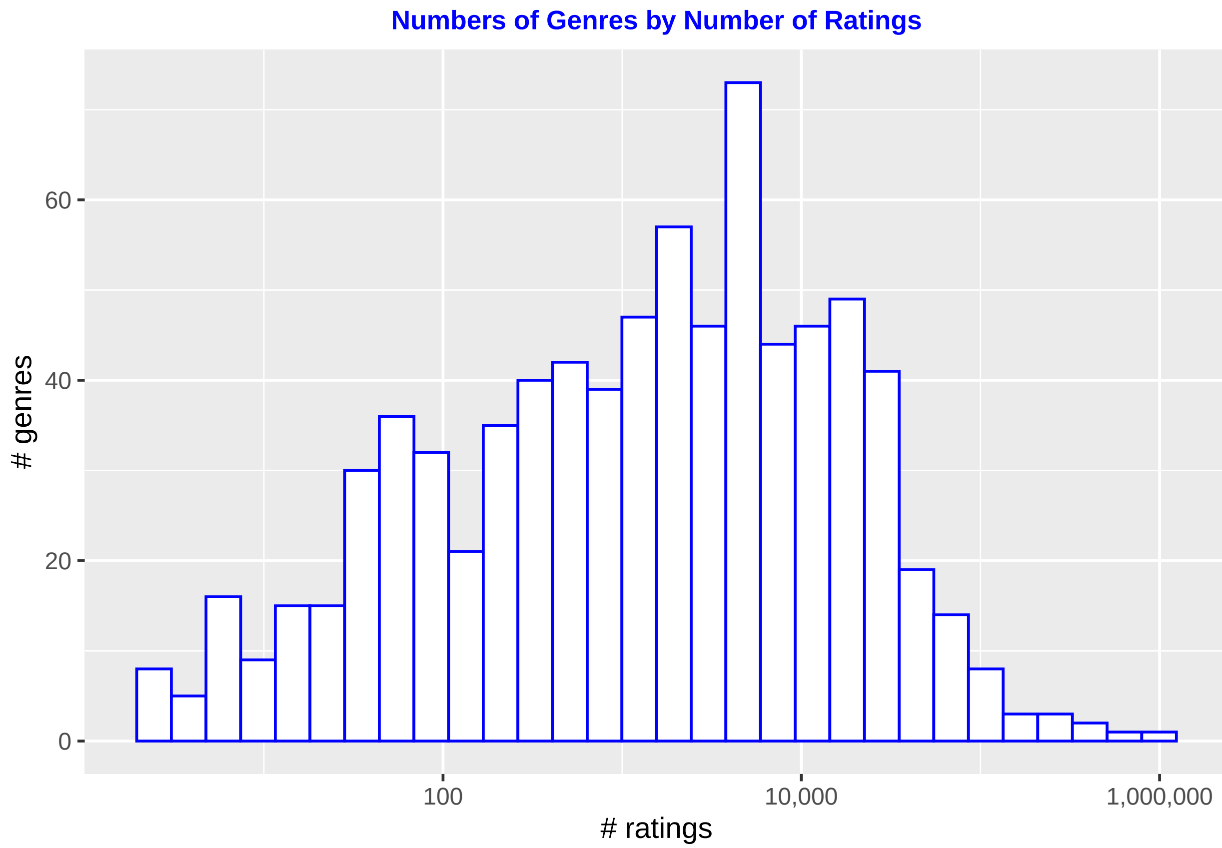
```
movielens %>% group_by(movieId) %>% summarize(count = n()) %>%
  ggplot(aes(count)) + geom_histogram(color = "blue", fill="white") +
  xlab("# ratings") + ylab("# movies") + ggtitle("Numbers of Movies by Number of Ratings") +
  theme(plot.title = element_text(hjust = 0.5, size = 10, color = "blue", face = "bold")) +
  scale_x_log10(labels = scales::comma)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



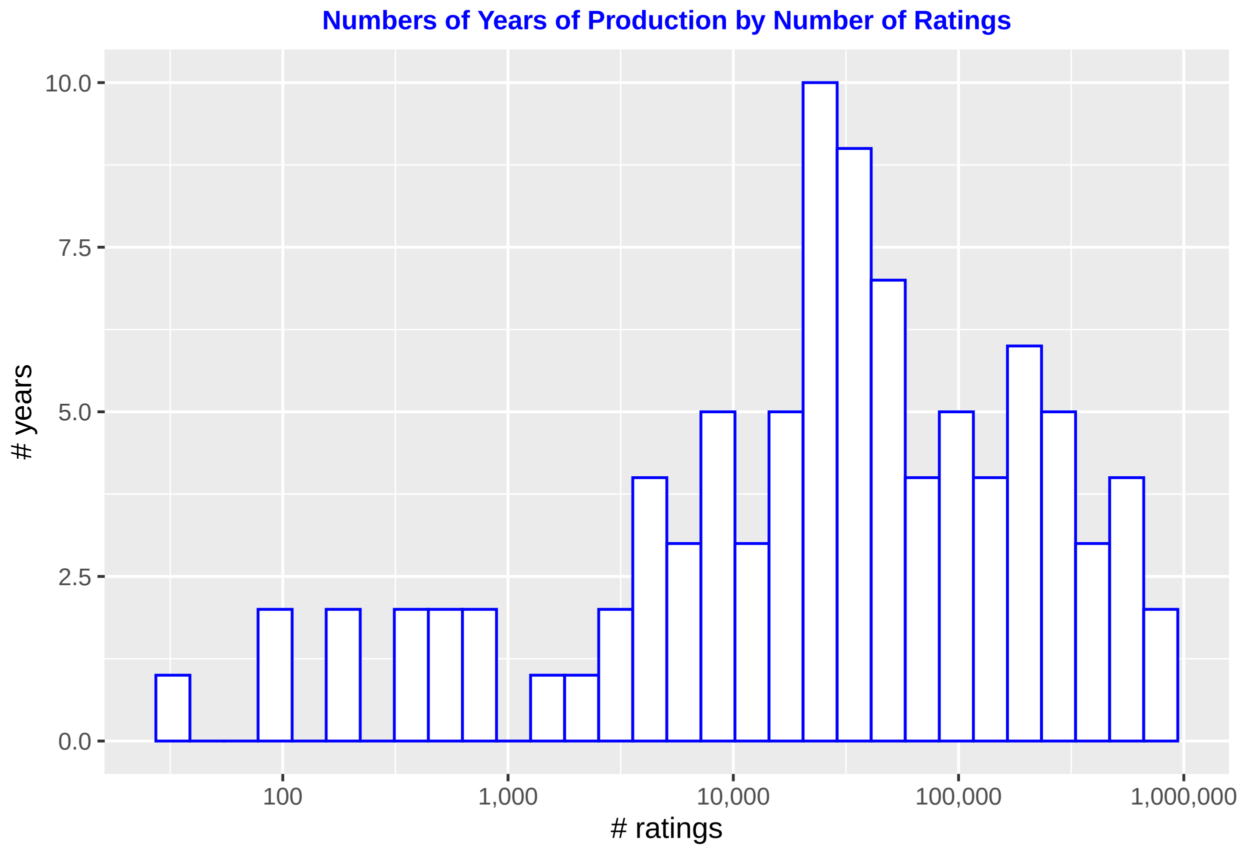
Plot Number of Genres by Number of Ratings

```
movielens %>% group_by(genres) %>% summarize(count = n()) %>%
  ggplot(aes(count)) + geom_histogram(color = "blue", fill="white") +
  xlab("# ratings") + ylab("# genres") + ggtitle("Numbers of Genres by Number of Ratings") +
  theme(plot.title = element_text(hjust = 0.5, size = 10, color = "blue", face = "bold")) +
  scale_x_log10(labels = scales::comma)
```



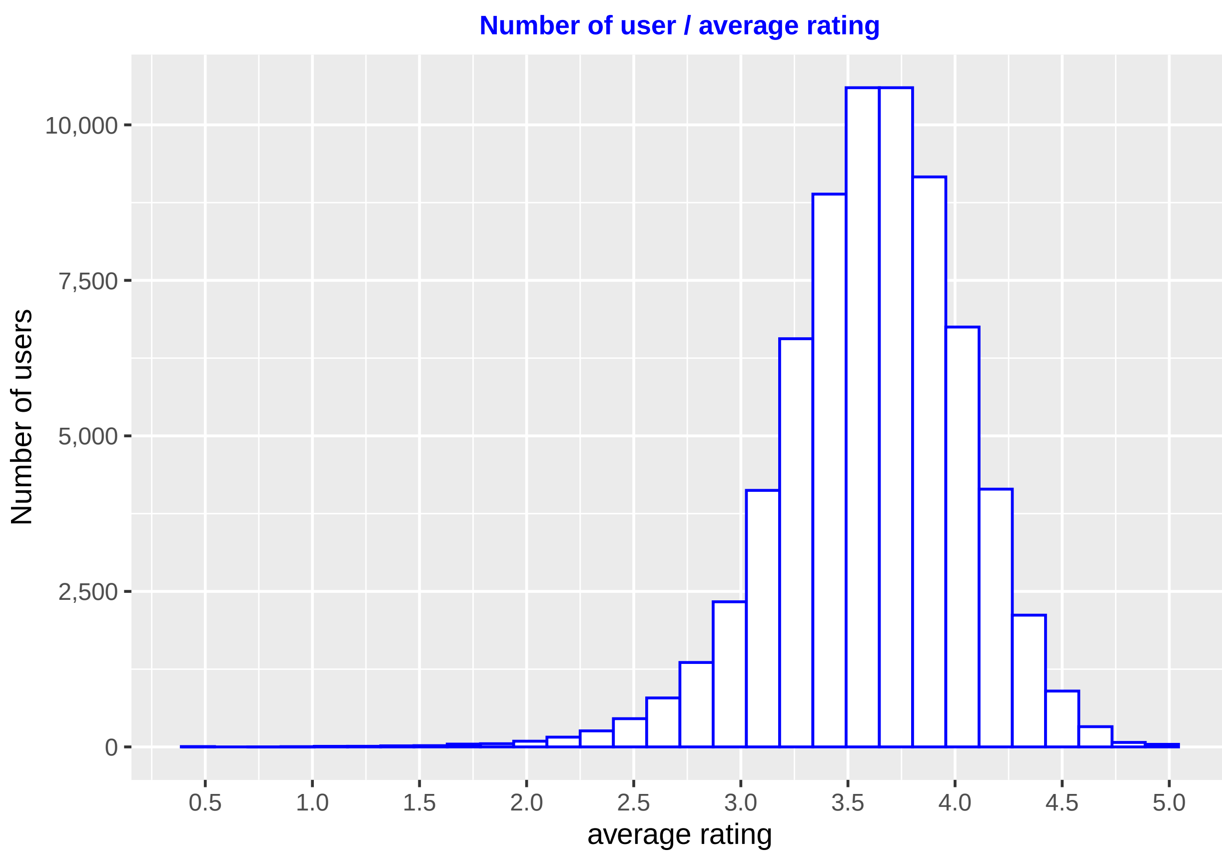
Plot Number of Years of Production by Number of Ratings

```
movielens %>% group_by(year) %>% summarize(count = n()) %>%
  ggplot(aes(count)) + geom_histogram(color = "blue", fill="white") +
  xlab("# ratings") + ylab("# years") + ggtitle("Numbers of Years of Production by Number of Ratings") +
  theme(plot.title = element_text(hjust = 0.5, size = 10, color = "blue", face = "bold")) +
  scale_x_log10(labels = scales::comma)
```



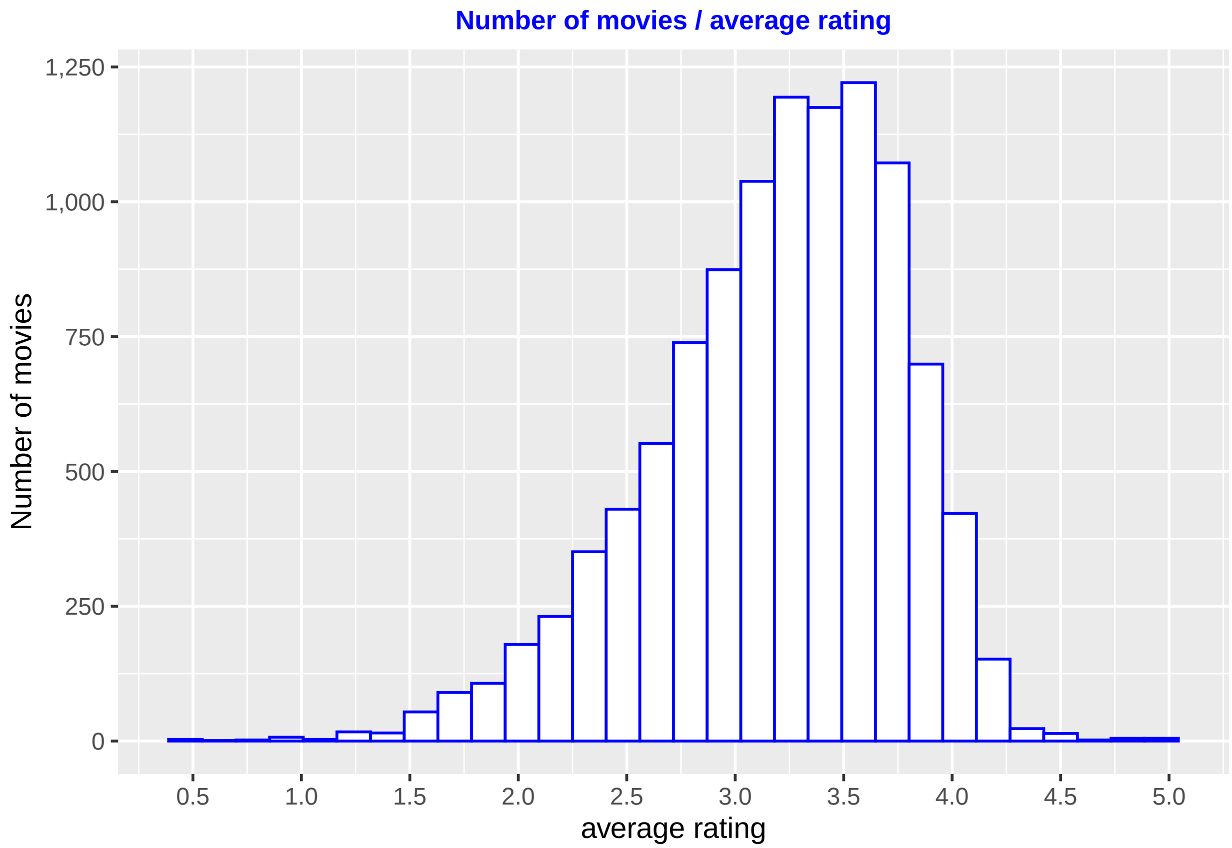
Plot of Number of Users / Average Rating

```
movielens %>% group_by(userId) %>%
  summarize(avg_rating = mean(rating)) %>%
  ggplot(aes(avg_rating)) +
  geom_histogram(color="blue", fill="white") +
  xlab("average rating") + ylab("Number of users") +
  ggtitle("Number of user / average rating") +
  theme(plot.title = element_text(hjust = 0.5, size = 10, color = "blue", face = "bold")) +
  scale_x_continuous(breaks = seq(0,5,0.5)) +
  scale_y_continuous(labels = scales::comma)
```



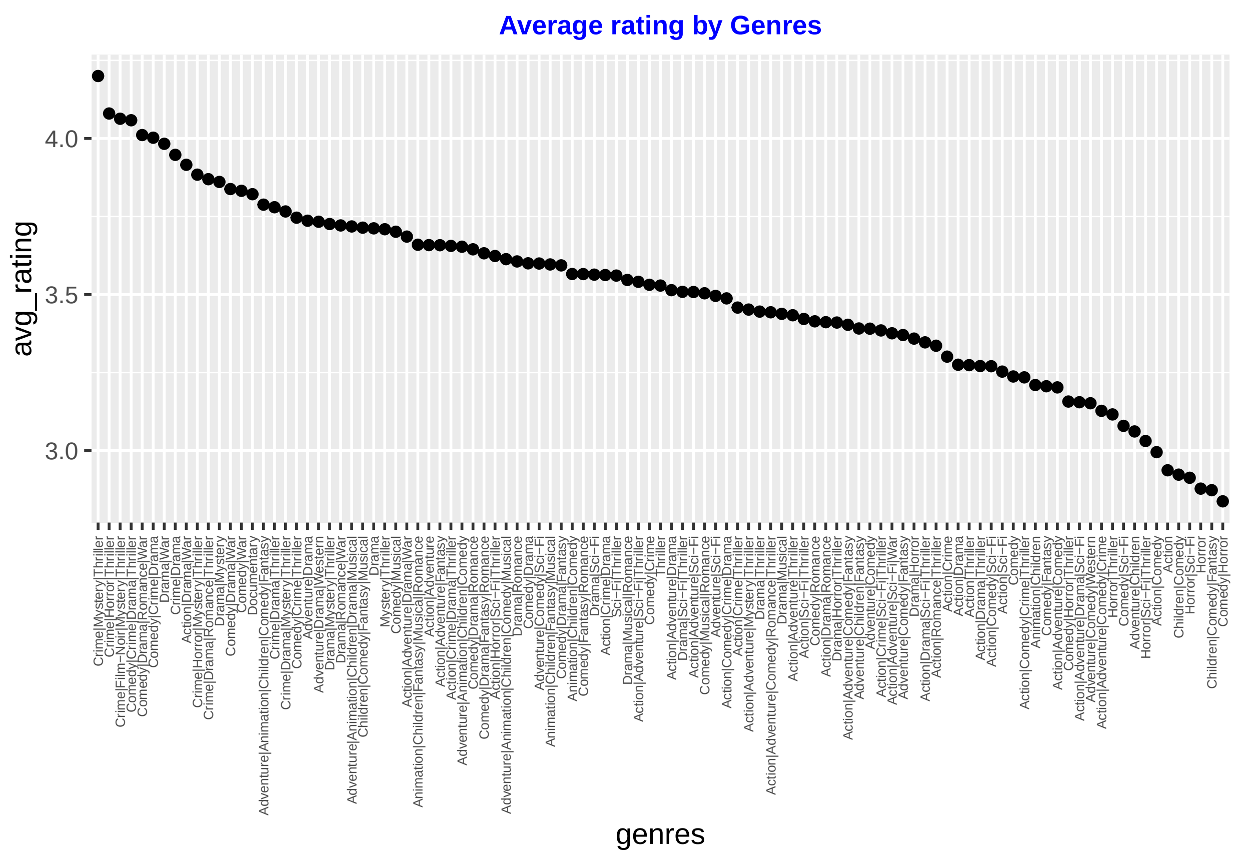
Plot of Number of Movies / Average ratings

```
movielens %>% group_by(movieId) %>%
  summarize(avg_rating = mean(rating)) %>%
  ggplot(aes(avg_rating)) +
  geom_histogram(color="blue", fill="white") +
  xlab("average rating") + ylab("Number of movies") +
  ggtitle("Number of movies / average rating") +
  theme(plot.title = element_text(hjust = 0.5, size = 10, color = "blue", face = "bold")) +
  scale_x_continuous(breaks = seq(0,5,0.5)) +
  scale_y_continuous(labels = scales::comma)
```



Plot average ratings by genres
We observe that some genres received high rating while other get low rating

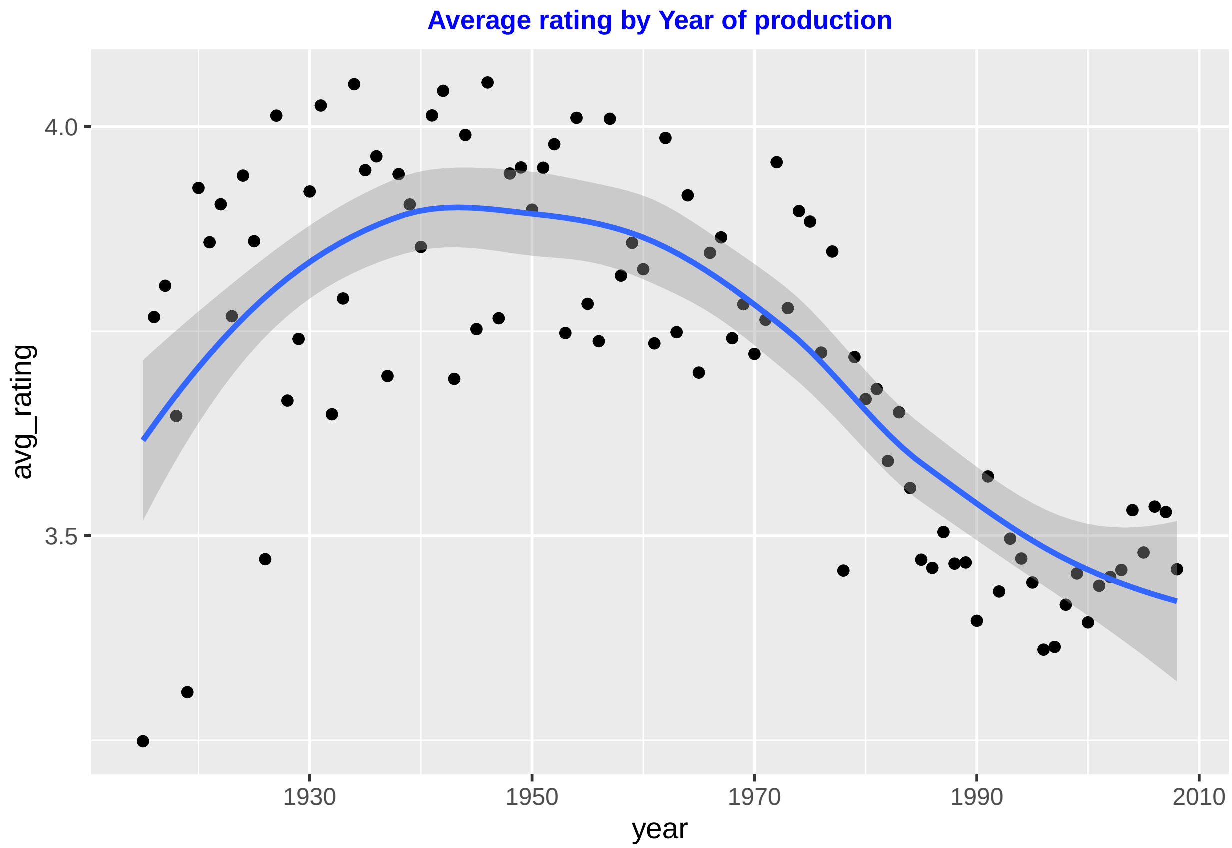
```
movielens %>% group_by(genres) %>%
  summarize(n = n(), avg_rating = mean(rating)) %>%
  filter(n > 20000) %>%
  mutate(genres = reorder(genres, desc(avg_rating))) %>%
  ggplot(aes(genres, avg_rating)) +
  geom_point() + geom_smooth() +
  ggtitle("Average rating by Genres") +
  theme(plot.title = element_text(hjust = 0.5, size = 10, color = "blue", face = "bold"),
        axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5, size = 5)) +
  scale_y_continuous(breaks = seq(0,5,0.5))
```



Plot average ratings by year of production

We observe that at some movie production years, the rating were given high while others were not.

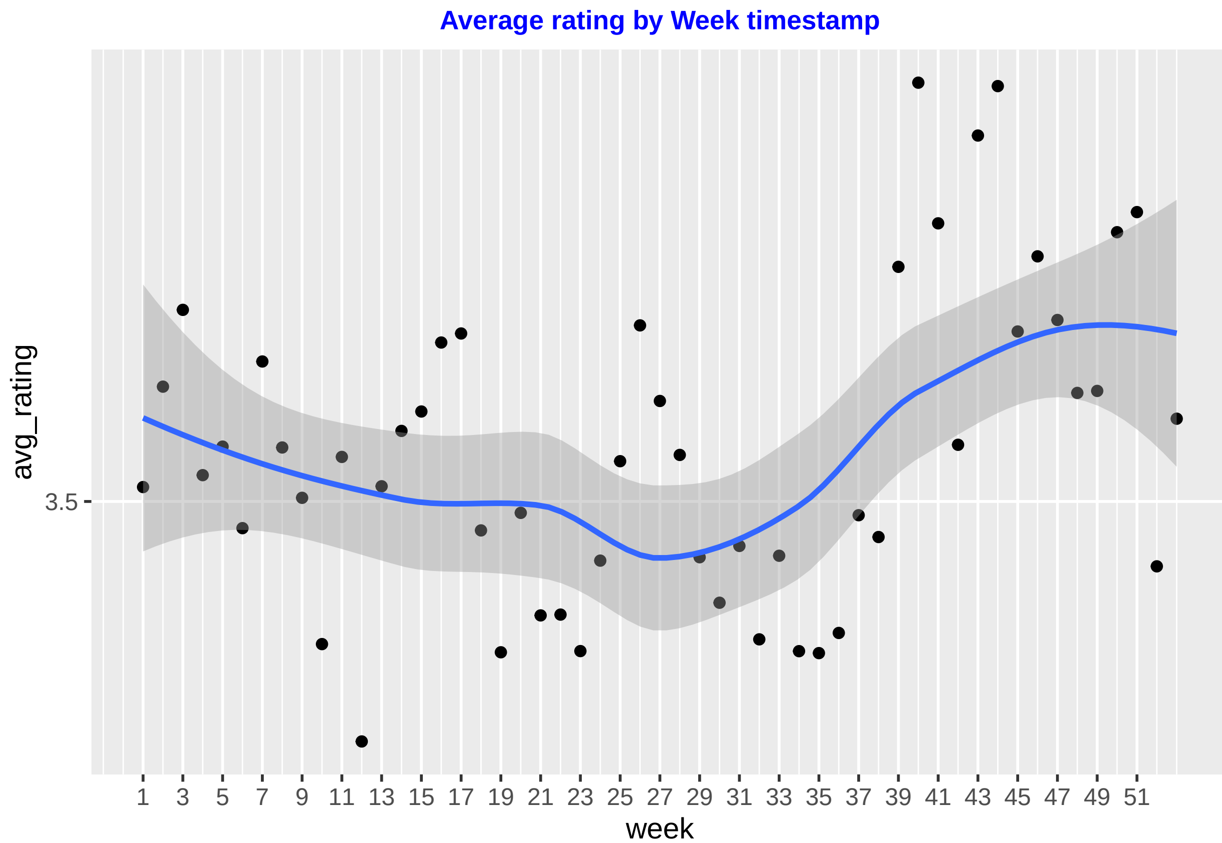
```
movielens %>% group_by(year) %>%
  summarize(avg_rating = mean(rating)) %>%
  ggplot(aes(year, avg_rating)) +
  geom_point() + geom_smooth() +
  ggtitle("Average rating by Year of production") +
  theme(plot.title = element_text(hjust = 0.5, size = 10, color = "blue", face = "bold")) +
  scale_y_continuous(breaks = seq(0,5,0.5))
```



Plot average ratings by week timestamp

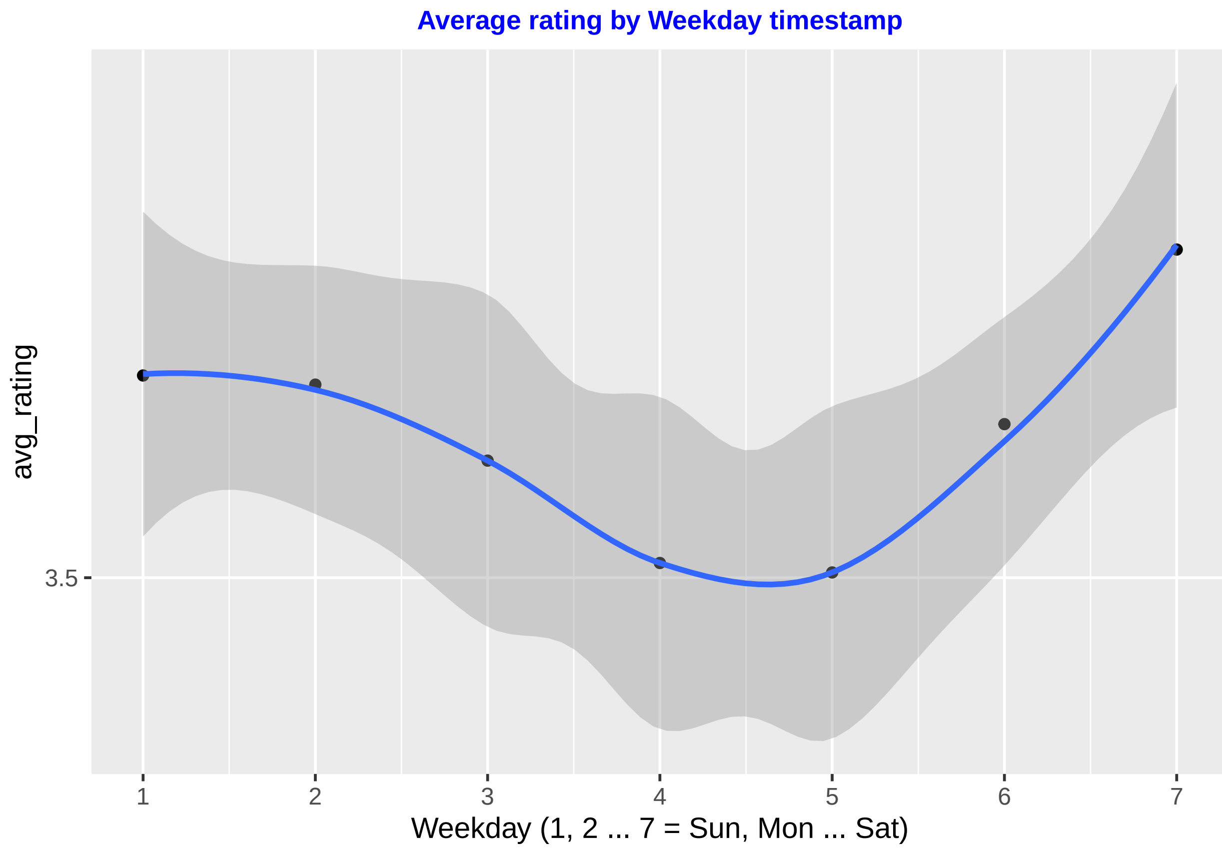
We observe that at the weeks at around the beginning and the end of the year, user give a little higher rating than those weeks in the middle of the year.

```
movielens %>% group_by(week) %>%
  summarize(avg_rating = mean(rating)) %>%
  ggplot(aes(week, avg_rating)) +
  geom_point() + geom_smooth() +
  ggtitle("Average rating by Week timestamp") +
  theme(plot.title = element_text(hjust = 0.5, size = 10, color = "blue", face = "bold")) +
  scale_y_continuous(breaks = seq(0,5,0.5)) +
  scale_x_continuous(breaks = seq(1,52,2))
```



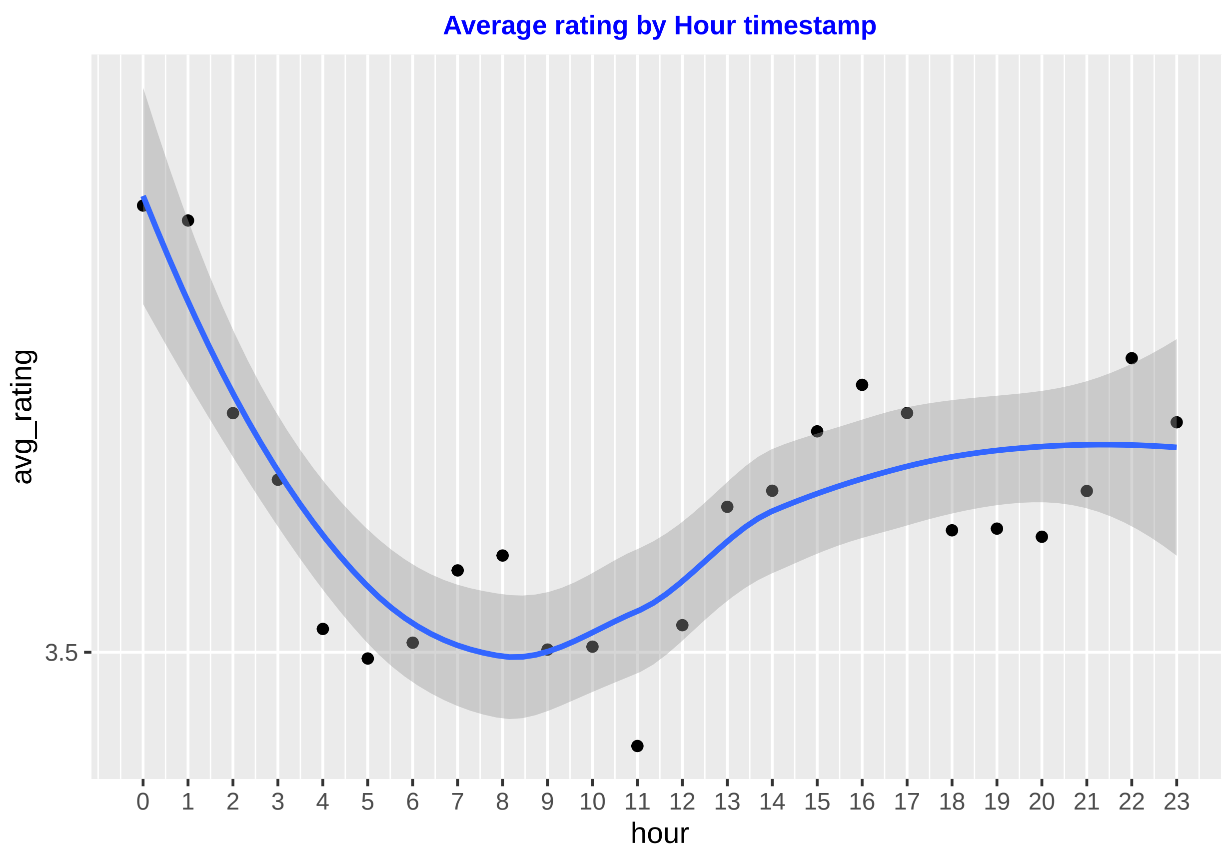
Plot average ratings by weekday timestamp
We observe that user give high rating on weekend vs. low rating on Wed and Thu.

```
movielens %>% group_by(wday) %>%
    summarize(avg_rating = mean(rating)) %>%
    ggplot(aes(wday, avg_rating)) +
    geom_point() + geom_smooth() +
    xlab("Weekday (1, 2 ... 7 = Sun, Mon ... Sat)") +
    ggtitle("Average rating by Weekday timestamp") +
    theme(plot.title = element_text(hjust = 0.5, size = 10, color = "blue", face = "bold")) +
    scale_y_continuous(breaks = seq(0,5,0.5)) +
    scale_x_continuous(breaks = seq(1,7,1))
```



Plot average ratings by hour timestamp
We observe that users giving high rating at around midnight while lower rating at around 7am to 9am

```
movielens %>% group_by(hour) %>%
  summarize(avg_rating = mean(rating)) %>%
  ggplot(aes(hour, avg_rating)) +
  geom_point() + geom_smooth() +
  ggtitle("Average rating by Hour timestamp") +
  theme(plot.title = element_text(hjust = 0.5, size = 10, color = "blue", face = "bold")) +
  scale_y_continuous(breaks = seq(0,5,0.5)) +
  scale_x_continuous(breaks = seq(0,23,1))
```



From observation above, we see that beside userId and movieId, other features also have relationship with rating value: genres, year of production and time that rating was given. We will take into account all of these features in our prediction algorithms.

Partitioning dataset

Partition Movielens dataset into **edx** (90%) and **validation** (10%) and then partition **edx** into **train_set** (90%) and **test_set** (10%) for further processing

```
# Validation set will be 10% of MovieLens data
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,]

# 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)


# Create training and testing data from edx dataset to build model
set.seed(1, sample.kind="Rounding")
```

```r
# Partition edx datase to training and testing data
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)

train_set <- edx[-test_index]
temp <- edx[test_index]

# Make sure userId and movieId in test_set are also in train_set
test_set <- temp %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")

# Add rows removed from test set back into train set
removed <- anti_join(temp, test_set)
train_set <- rbind(train_set, removed)

# Remove unused variables
rm(dl, ratings, movies, test_index, temp, removed)
```

Crosscheck number of instances in each sub-dataset and its percentage: we see that **validation** dataset is 10% of **movielens** dataset and **test_set** is 10% of **edx** dataset.

```r
data.frame(movielens = c(nrow(movielens), 100),
           edx = c(nrow(edx), round(nrow(edx)/nrow(movielens) * 100)),
           validation = c(nrow(validation), round(nrow(validation) / nrow(movielens)*100)),
           row.names = c("Number of instances", "Percentage (%)")) %>% knitr::kable()
```

	movielens	edx	validation
Number of instances	10000054	9000055	999999
Percentage (%)	100	90	10

```r
data.frame(edx = c(nrow(edx), 100),
           train_set = c(nrow(train_set), round(nrow(train_set)/nrow(edx) * 100)),
           test_set = c(nrow(test_set), round(nrow(test_set) / nrow(edx)*100)),
           row.names = c("Number of instances", "Percentage (%)")) %>% knitr::kable()
```

	edx	train_set	test_set
Number of instances	9000055	8100065	899990
Percentage (%)	100	90	10

Evaluate RMSE of different algorithms on train_set and test_set dataset

Define RMSE function to calculate RMSE between predictions and actual rating

```r
RMSE <- function(real_rating, predicted_rating) {
  sqrt(mean((real_rating - predicted_rating)^2))
}
```

1. Average prediction model

```r
predicted_ratings <- mean(train_set$rating)

rmses_score <- data.frame(method = "Average Rating",
                          rmse_score= RMSE(test_set$rating, predicted_ratings))

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

method	rmse_score
Average Rating	1.060054

2. Movie model

```r
mu <- mean(train_set$rating)
b_i <- train_set %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
predicted_ratings <- test_set %>%
  left_join(b_i, by = "movieId") %>%
  mutate(pred = mu + b_i) %>% .$pred

rmses_score <- rbind(rmses_score, c("Movie",
                                    RMSE(test_set$rating, predicted_ratings)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501

3. Movie + User model

```r
b_u <- train_set %>%
  left_join(b_i, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

predicted_ratings <- test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>% .$pred

rmses_score <- rbind(rmses_score, c("Movie + User",
                                    RMSE(test_set$rating, predicted_ratings)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501

method	rmse_score
Movie + User	0.86468429490229

4. Movie + User + Genres model

```
b_g <- train_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu - b_i - b_u))

predicted_ratings <- test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  mutate(pred = mu + b_i + b_u + b_g) %>% .$pred

rmses_score <- rbind(rmses_score, c("Movie + User + Genres",
                                    RMSE(test_set$rating, predicted_ratings)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501
Movie + User	0.86468429490229
Movie + User + Genres	0.864324145155978

5. Movie + User + Genres + Production Year model

```
b_y <- train_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  group_by(year) %>%
  summarize(b_y = mean(rating - mu - b_i - b_u - b_g))

predicted_ratings <- test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  left_join(b_y, by = "year") %>%
  mutate(pred = mu + b_i + b_u + b_g + b_y) %>% .$pred

rmses_score <- rbind(rmses_score, c("Movie/User/Genres/Production Year",
                                    RMSE(test_set$rating, predicted_ratings)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501
Movie + User	0.86468429490229
Movie + User + Genres	0.864324145155978
Movie/User/Genres/Production Year	0.864126155252257

6. Movie + User + Genres + Production Year + Week model

```
b_w <- train_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  left_join(b_y, by = "year") %>%
  group_by(week) %>%
  summarize(b_w = mean(rating - mu - b_i - b_u - b_g + b_y))

predicted_ratings <- test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_w, by = "week") %>%
  mutate(pred = mu + b_i + b_u + b_g + b_y + b_w) %>% .$pred

rmses_score <- rbind(rmses_score, c("Movie/User/Genres/Production Year/Week",
                                    RMSE(test_set$rating, predicted_ratings)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501
Movie + User	0.86468429490229
Movie + User + Genres	0.864324145155978
Movie/User/Genres/Production Year	0.864126155252257
Movie/User/Genres/Production Year/Week	0.864115951426679

7. Movie + User + Genres + Production Year + Week + Weekday model

```
b_wd <- train_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_w, by = "week") %>%
  group_by(wday) %>%
  summarize(b_wd = mean(rating - mu - b_i - b_u - b_g + b_y + b_w))

predicted_ratings <- test_set %>%
  left_join(b_i, by = "movieId") %>%
```

```r
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_w, by = "week") %>%
  left_join(b_wd, by = "wday") %>%
  mutate(pred = mu + b_i + b_u + b_g + b_y + b_w + b_wd) %>% .$pred

rmses_score <- rbind(rmses_score, c("Movie/User/Genres/Production Year/Week/Weekday",
                                    RMSE(test_set$rating, predicted_ratings)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501
Movie + User	0.86468429490229
Movie + User + Genres	0.864324145155978
Movie/User/Genres/Production Year	0.864126155252257
Movie/User/Genres/Production Year/Week	0.864115951426679
Movie/User/Genres/Production Year/Week/Weekday	0.864116475178773

8. Movie + User + Genres + Production Year + Week + Weekday + Hour model

```r
b_h <- train_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_w, by = "week") %>%
  left_join(b_wd, by = "wday") %>%
  group_by(hour) %>%
  summarize(b_h = mean(rating - mu - b_i - b_u - b_g + b_y + b_w + b_wd))

predicted_ratings <- test_set %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  left_join(b_y, by = "year") %>%
  left_join(b_w, by = "week") %>%
  left_join(b_wd, by = "wday") %>%
  left_join(b_h, by = "hour") %>%
  mutate(pred = mu + b_i + b_u + b_g + b_y + b_w + b_wd + b_h) %>% .$pred

rmses_score <- rbind(rmses_score, c("Movie/User/Genres/Production Year/Week/Weekday/Hour",
                                    RMSE(test_set$rating, predicted_ratings)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501
Movie + User	0.86468429490229

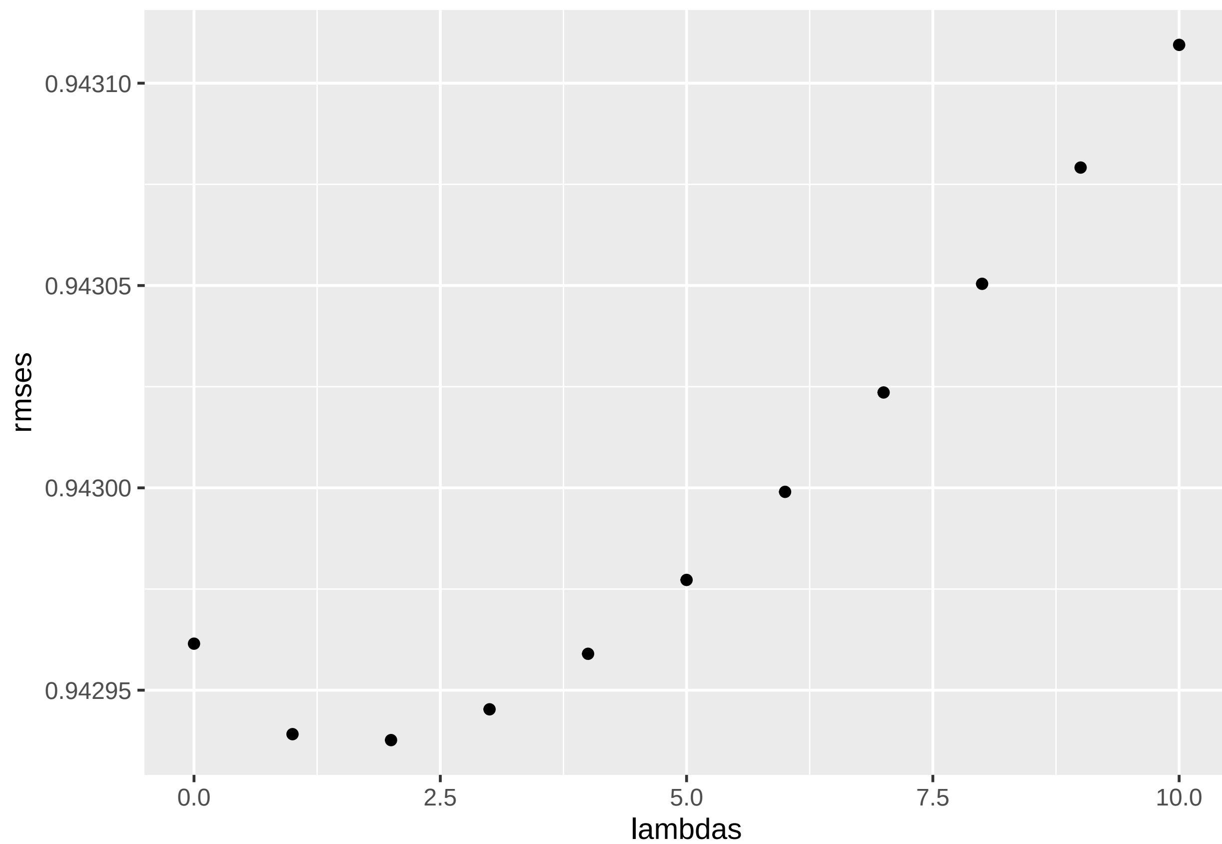
method	rmse_score
Movie + User + Genres	0.864324145155978
Movie/User/Genres/Production Year	0.864126155252257
Movie/User/Genres/Production Year/Week	0.864115951426679
Movie/User/Genres/Production Year/Week/Weekday	0.864116475178773
Movie/User/Genres/Production Year/Week/Weekday/Hour	0.864116872117947

9. Regularized Movie

```r
lambdas <- seq(0,10,1)
mu <- mean(train_set$rating)

rmses <- sapply(lambdas, function(l) {
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + l))
  predicted_ratings <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    mutate(pred = mu + b_i) %>%
    .$pred
  RMSE(test_set$rating, predicted_ratings)
})

rmses_score <- rbind(rmses_score,
                     c(paste("Reg. Movie (lamda = ",lambdas[which.min(rmses)], ")",
                             sep = ""), min(rmses)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501
Movie + User	0.86468429490229
Movie + User + Genres	0.864324145155978
Movie/User/Genres/Production Year	0.864126155252257
Movie/User/Genres/Production Year/Week	0.864115951426679
Movie/User/Genres/Production Year/Week/Weekday	0.864116475178773
Movie/User/Genres/Production Year/Week/Weekday/Hour	0.864116872117947
Reg. Movie (lamda = 2)	0.942937666884635

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

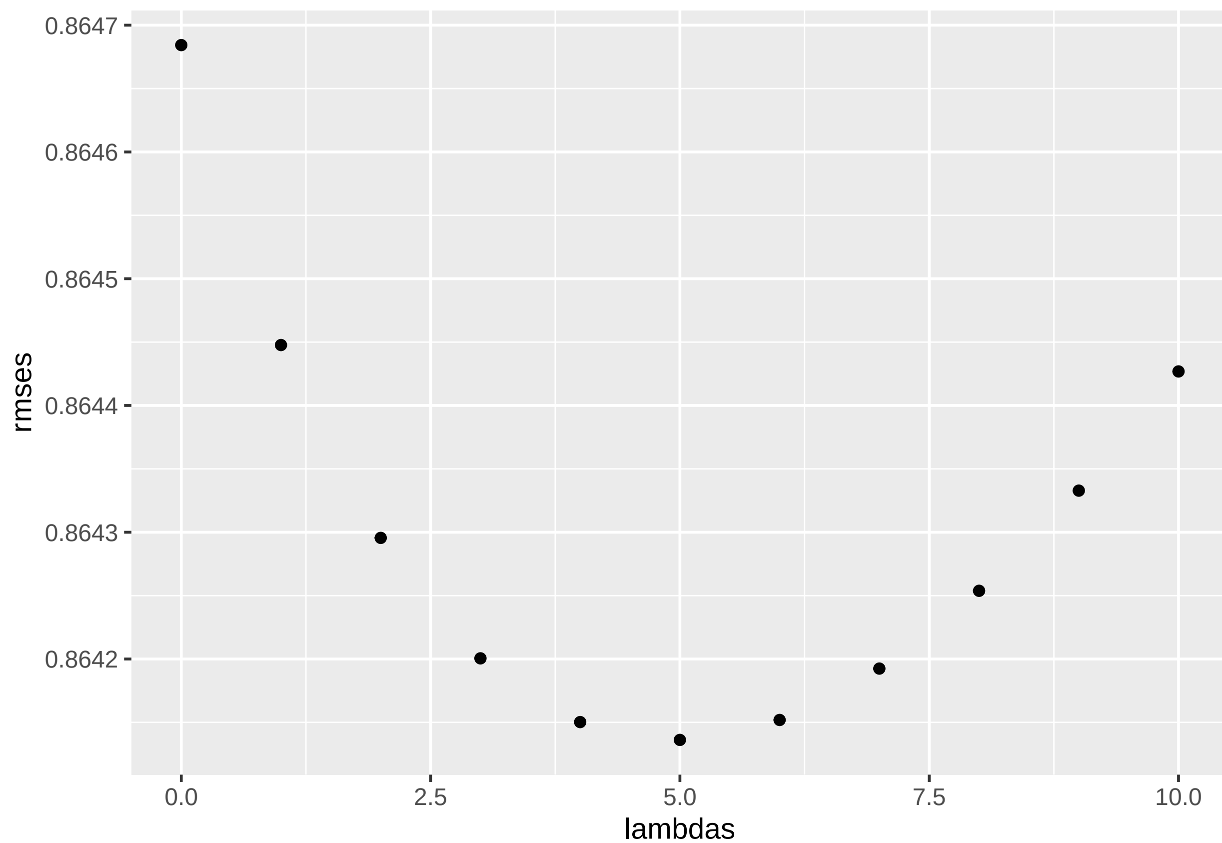


10. Regularized Movie + User

```
rmses <- sapply(lambdas, function(l) {
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + l))
  b_u <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_i)/(n() + l))

  predicted_ratings <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred
  RMSE(test_set$rating, predicted_ratings)
})

rmses_score <- rbind(rmses_score,
                     c(paste("Reg. Movie + User (lamda = ",lambdas[which.min(rmses)], ")",
                             sep = ""), min(rmses)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501
Movie + User	0.86468429490229
Movie + User + Genres	0.864324145155978
Movie/User/Genres/Production Year	0.864126155252257
Movie/User/Genres/Production Year/Week	0.864115951426679
Movie/User/Genres/Production Year/Week/Weekday	0.864116475178773
Movie/User/Genres/Production Year/Week/Weekday/Hour	0.864116872117947
Reg. Movie (lamda = 2)	0.942937666884635
Reg. Movie + User (lamda = 5)	0.864136179290374

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



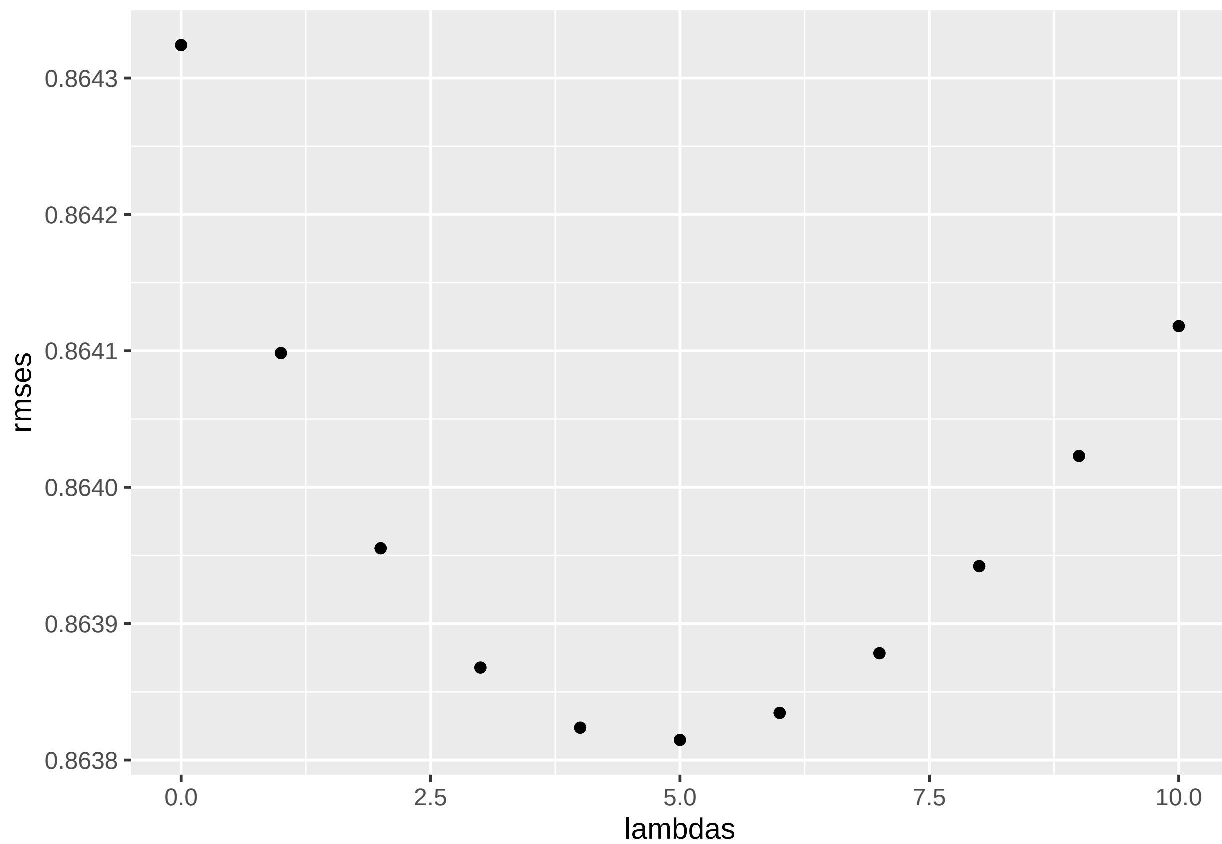
11. Regularized Movie + User + Genres

```
rmses <- sapply(lambdas, function(l) {
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + l))
  b_u <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_i)/(n() + l))
```

```
  b_g <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u)/(n() + l))

  predicted_ratings <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    mutate(pred = mu + b_i + b_u + b_g) %>%
    .$pred
  RMSE(test_set$rating, predicted_ratings)
})

rmses_score <- rbind(rmses_score,
                     c(paste("Reg. Movie + User + Genres (lamda = ",
                             lambdas[which.min(rmses)], ")", sep = ""), min(rmses)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501
Movie + User	0.86468429490229
Movie + User + Genres	0.864324145155978
Movie/User/Genres/Production Year	0.864126155252257
Movie/User/Genres/Production Year/Week	0.864115951426679
Movie/User/Genres/Production Year/Week/Weekday	0.864116475178773
Movie/User/Genres/Production Year/Week/Weekday/Hour	0.864116872117947
Reg. Movie (lamda = 2)	0.942937666884635
Reg. Movie + User (lamda = 5)	0.864136179290374
Reg. Movie + User + Genres (lamda = 5)	0.863814759978585

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



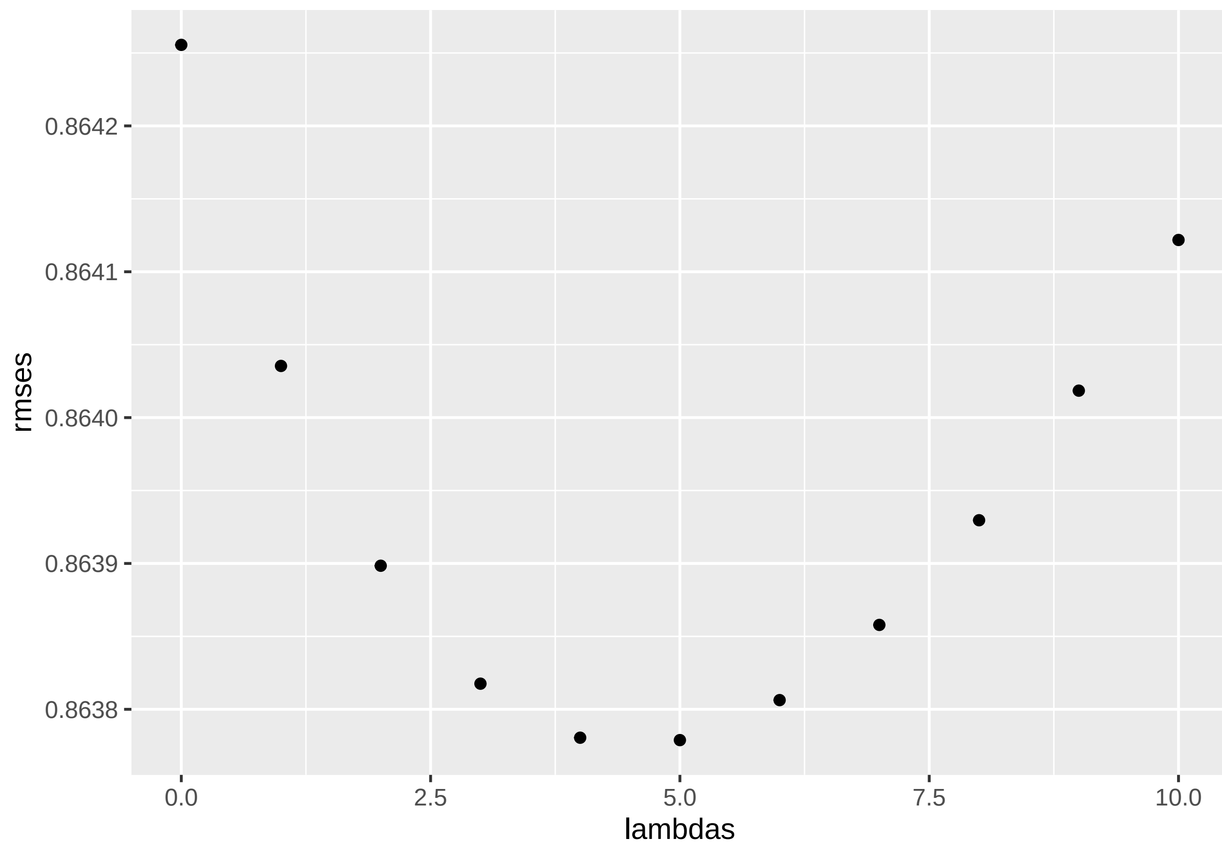
12. Regularized Movie + User + Genres + Production Year

```
rmses <- sapply(lambdas, function(l) {
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + l))
  b_u <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_i)/(n() + l))
  b_g <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u)/(n() + l))
  b_y <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    group_by(year) %>%
    summarize(b_y = sum(rating - mu - b_i - b_u + b_g)/(n() + l))

  predicted_ratings <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
```

```
    left_join(b_y, by = "year") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_y) %>%
    .$pred
  RMSE(test_set$rating, predicted_ratings)
})

rmses_score <- rbind(rmses_score,
                     c(paste("Reg. Movie/User/Genres/Production Year (lamda=",
                             lambdas[which.min(rmses)], ")", sep = ""),
                       min(rmses)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501
Movie + User	0.86468429490229
Movie + User + Genres	0.864324145155978
Movie/User/Genres/Production Year	0.864126155252257
Movie/User/Genres/Production Year/Week	0.864115951426679
Movie/User/Genres/Production Year/Week/Weekday	0.864116475178773
Movie/User/Genres/Production Year/Week/Weekday/Hour	0.864116872117947
Reg. Movie (lamda = 2)	0.942937666884635
Reg. Movie + User (lamda = 5)	0.864136179290374
Reg. Movie + User + Genres (lamda = 5)	0.863814759978585
Reg. Movie/User/Genres/Production Year (lamda=5)	0.863778893988623

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



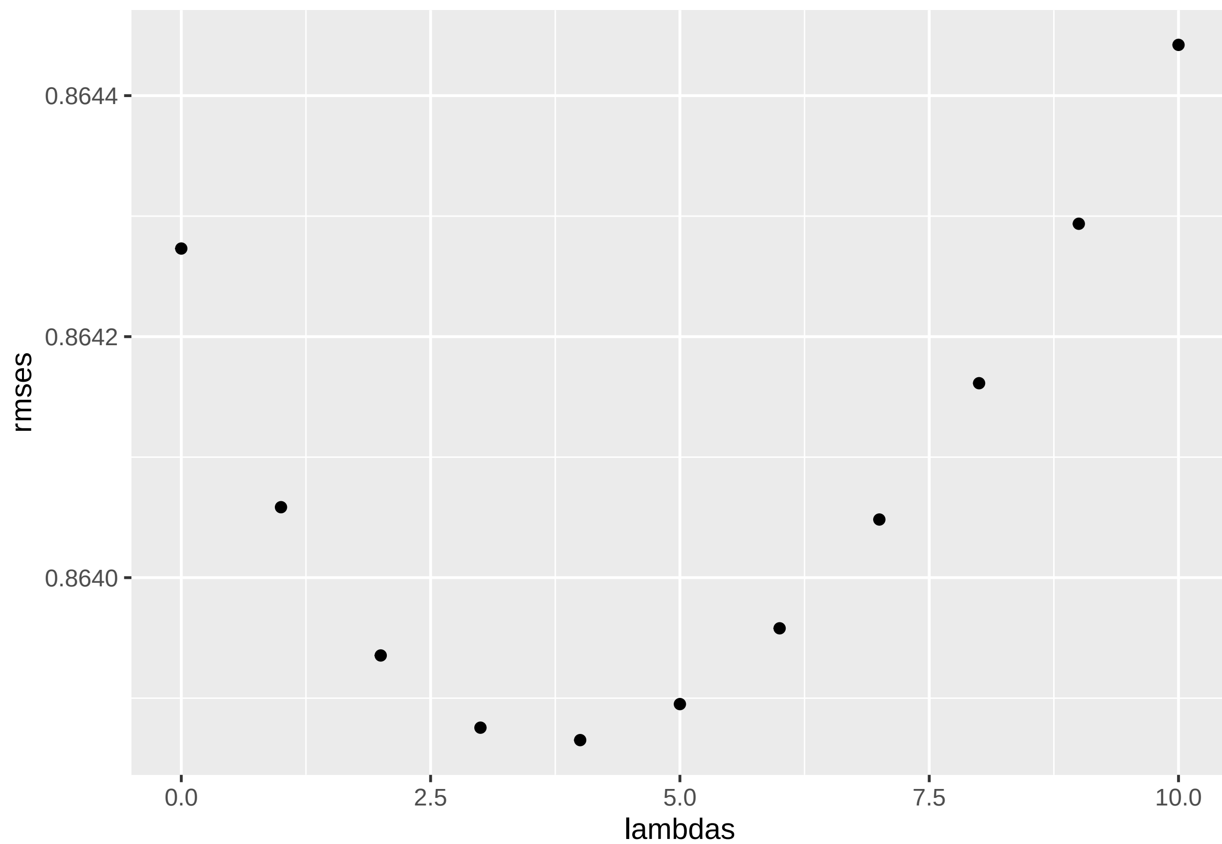
13. Regularized Movie + User + Genres + Production Year + Week

```
rmses <- sapply(lambdas, function(l) {
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + l))
  b_u <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_i)/(n() + l))
  b_g <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u)/(n() + l))
  b_y <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    group_by(year) %>%
    summarize(b_y = sum(rating - mu - b_i - b_u + b_g)/(n() + l))
  b_w <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_y, by = "year") %>%
```

```
    group_by(week) %>%
    summarize(b_w = sum(rating - mu - b_i - b_u + b_g + b_y)/(n() + l))

  predicted_ratings <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_y, by = "year") %>%
    left_join(b_w, by = "week") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_y + b_w) %>%
    .$pred
  RMSE(test_set$rating, predicted_ratings)
})

rmses_score <- rbind(rmses_score,
                     c(paste("Reg. Movie/User/Genres/Production Year/Week (lamda=",
                             lambdas[which.min(rmses)], ")", sep = ""),
                       min(rmses)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501
Movie + User	0.86468429490229
Movie + User + Genres	0.864324145155978
Movie/User/Genres/Production Year	0.864126155252257
Movie/User/Genres/Production Year/Week	0.864115951426679
Movie/User/Genres/Production Year/Week/Weekday	0.864116475178773
Movie/User/Genres/Production Year/Week/Weekday/Hour	0.864116872117947
Reg. Movie (lamda = 2)	0.942937666884635
Reg. Movie + User (lamda = 5)	0.864136179290374
Reg. Movie + User + Genres (lamda = 5)	0.863814759978585
Reg. Movie/User/Genres/Production Year (lamda=5)	0.863778893988623
Reg. Movie/User/Genres/Production Year/Week (lamda=4)	0.863865201384017

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



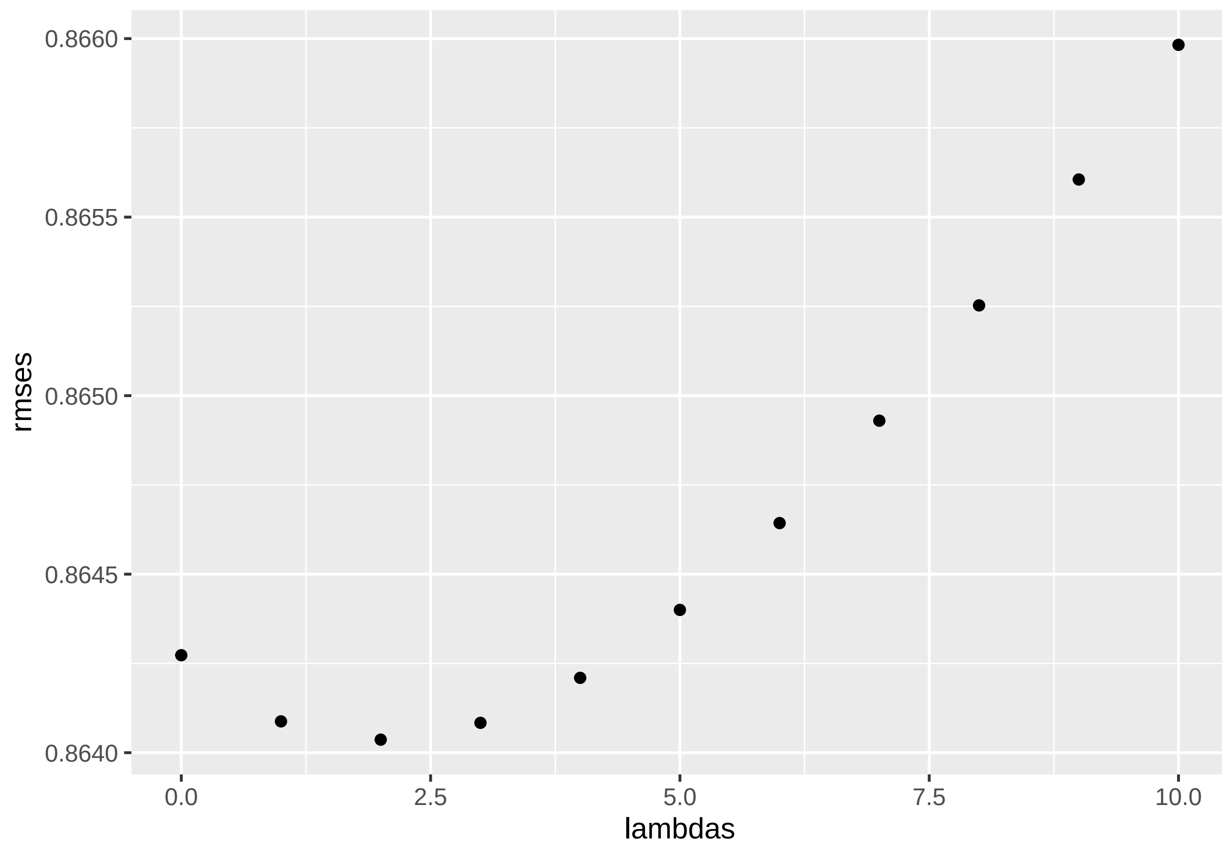
14. Regularized Movie + User + Genres + Production Year + Week + Weekday

```
rmses <- sapply(lambdas, function(l) {
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + l))
  b_u <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_i)/(n() + l))
  b_g <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u)/(n() + l))
  b_y <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    group_by(year) %>%
    summarize(b_y = sum(rating - mu - b_i - b_u + b_g)/(n() + l))
  b_w <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_y, by = "year") %>%
```

```r
    group_by(week) %>%
    summarize(b_w = sum(rating - mu - b_i - b_u + b_g + b_y)/(n() + l))
  b_wd <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_y, by = "year") %>%
    left_join(b_w, by = "week") %>%
    group_by(wday) %>%
    summarize(b_wd = sum(rating - mu - b_i - b_u + b_g + b_y + b_w)/(n() + l))

  predicted_ratings <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_y, by = "year") %>%
    left_join(b_w, by = "week") %>%
    left_join(b_wd, by = "wday") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_y + b_w + b_wd) %>%
    .$pred
  RMSE(test_set$rating, predicted_ratings)
})

rmses_score <- rbind(rmses_score,
                     c(paste("Reg. Movie/User/Genres/Production Year/Week/Weekday (lamda=",
                             lambdas[which.min(rmses)], ")", sep = ""),
                       min(rmses)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501
Movie + User	0.86468429490229
Movie + User + Genres	0.864324145155978
Movie/User/Genres/Production Year	0.864126155252257
Movie/User/Genres/Production Year/Week	0.864115951426679
Movie/User/Genres/Production Year/Week/Weekday	0.864116475178773
Movie/User/Genres/Production Year/Week/Weekday/Hour	0.864116872117947
Reg. Movie (lamda = 2)	0.942937666884635
Reg. Movie + User (lamda = 5)	0.864136179290374
Reg. Movie + User + Genres (lamda = 5)	0.863814759978585
Reg. Movie/User/Genres/Production Year (lamda=5)	0.863778893988623
Reg. Movie/User/Genres/Production Year/Week (lamda=4)	0.863865201384017
Reg. Movie/User/Genres/Production Year/Week/Weekday (lamda=2)	0.864036513152013

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



15. Regularized Movie + User + Genres + Production Year + Week + Weekday + Hour

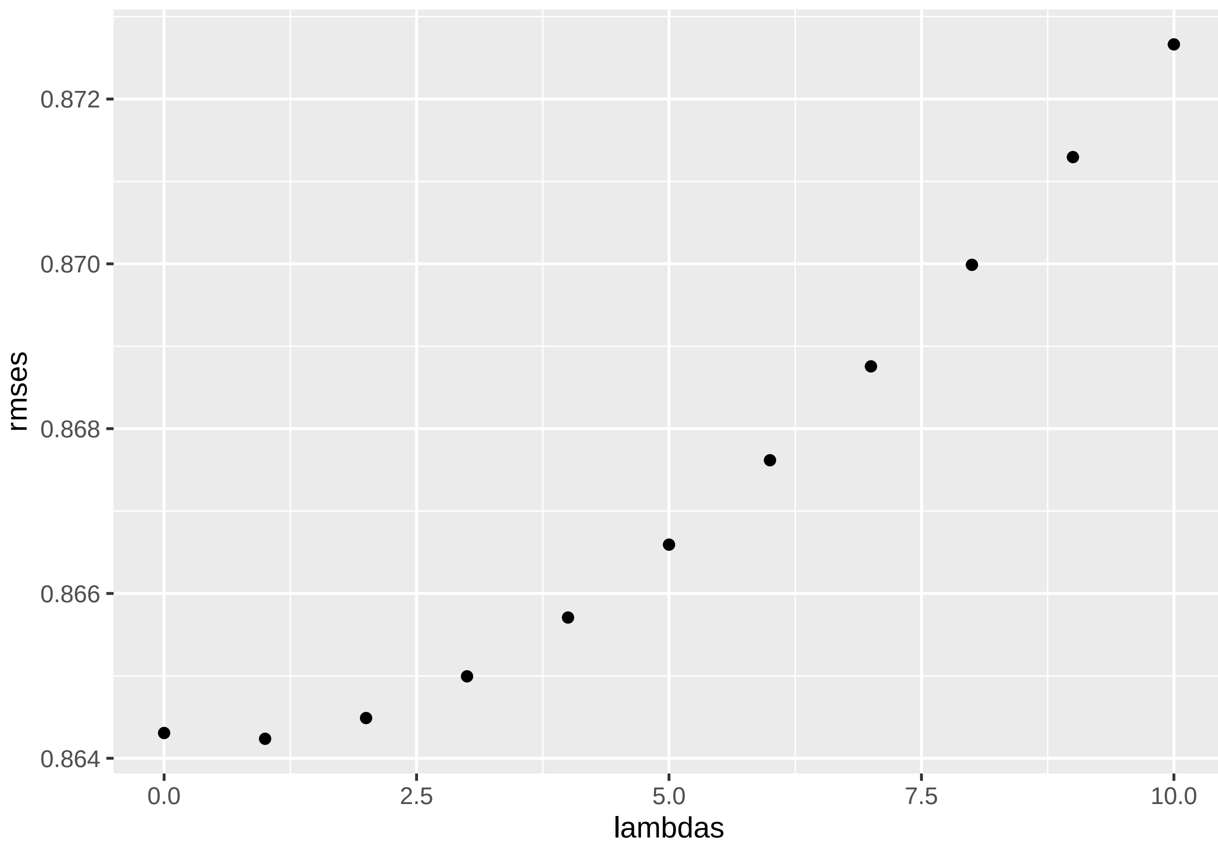
```
rmses <- sapply(lambdas, function(l) {
  b_i <- train_set %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n() + l))
  b_u <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - mu - b_i)/(n() + l))
  b_g <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    group_by(genres) %>%
    summarize(b_g = sum(rating - mu - b_i - b_u)/(n() + l))
  b_y <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    group_by(year) %>%
    summarize(b_y = sum(rating - mu - b_i - b_u + b_g)/(n() + l))
  b_w <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_y, by = "year") %>%
```

```r
    group_by(week) %>%
    summarize(b_w = sum(rating - mu - b_i - b_u + b_g + b_y)/(n() + 1))
  b_wd <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_y, by = "year") %>%
    left_join(b_w, by = "week") %>%
    group_by(wday) %>%
    summarize(b_wd = sum(rating - mu - b_i - b_u + b_g + b_y + b_w)/(n() + 1))
  b_h <- train_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_y, by = "year") %>%
    left_join(b_w, by = "week") %>%
    left_join(b_wd, by = "wday") %>%
    group_by(hour) %>%
    summarize(b_h = sum(rating - mu - b_i - b_u + b_g + b_y + b_w + b_wd)/(n() + 1))

  predicted_ratings <- test_set %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(b_g, by = "genres") %>%
    left_join(b_y, by = "year") %>%
    left_join(b_w, by = "week") %>%
    left_join(b_wd, by = "wday") %>%
    left_join(b_h, by = "hour") %>%
    mutate(pred = mu + b_i + b_u + b_g + b_y + b_w + b_wd + b_h) %>%
    .$pred
  RMSE(test_set$rating, predicted_ratings)
})

rmses_score <- rbind(rmses_score,
                     c(paste("Reg. Movie/User/Genres/Production Year/Week/Weekday/Hour
                             (lamda=", lambdas[which.min(rmses)], ")", sep = ""),
                       min(rmses)))
rmses_score %>% knitr::kable()
```

method	rmse_score
Average Rating	1.06005370222409
Movie	0.942961498004501
Movie + User	0.86468429490229
Movie + User + Genres	0.864324145155978
Movie/User/Genres/Production Year	0.864126155252257
Movie/User/Genres/Production Year/Week	0.864115951426679
Movie/User/Genres/Production Year/Week/Weekday	0.864116475178773
Movie/User/Genres/Production Year/Week/Weekday/Hour	0.864116872117947
Reg. Movie (lamda = 2)	0.942937666884635
Reg. Movie + User (lamda = 5)	0.864136179290374
Reg. Movie + User + Genres (lamda = 5)	0.863814759978585
Reg. Movie/User/Genres/Production Year (lamda=5)	0.863778893988623

method	rmse_score
Reg. Movie/User/Genres/Production Year/Week (lamda=4)	0.863865201384017
Reg. Movie/User/Genres/Production Year/Week/Weekday (lamda=2)	0.864036513152013
Reg. Movie/User/Genres/Production Year/Week/Weekday/Hour (lamda=1)	0.864236434817133

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



The model that give minimum RMSE is:

```
# Show the Model that return minimum RMSE
rmses_score[which.min(rmses_score$rmse_score),] %>% knitr::kable()
```

	method	rmse_score
12	Reg. Movie/User/Genres/Production Year (lamda=5)	0.863778893988623

```
# the min RMSE model shown that lambda_min = 5
```

Apply best model on edx data and evaluate on validation data

From results above, we observe that Regularized of MovieId/UserId/Genres/Production Year model (at lambda = 5) give minimum RMSE. We will apply this model in **edx** data to predict rating of **validation** data and calculate the RMSE with actual rating in **validation** data.

```r
mu <- mean(edx$rating)
lambda_min <- 5

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

predicted_ratings <- validation %>%
  left_join(b_i, by = "movieId") %>%
  left_join(b_u, by = "userId") %>%
  left_join(b_g, by = "genres") %>%
  left_join(b_y, by = "year") %>%
  mutate(pred = mu + b_i + b_u + b_g + b_y) %>%
  .$pred

rmse <- RMSE(validation$rating, predicted_ratings)

print(paste("RMSE =", rmse, ", compare with target: RMSE < 0.86490 is", rmse < 0.86490))
```

```
## [1] "RMSE = 0.864408167515197 , compare with target: RMSE < 0.86490 is TRUE"
```

Results

RMSE of predicted rating vs. actual rating in **validation** dataset is 0.8644082

We achieved the target of this project.

END OF REPORT