Movielens Project Report

LH

10/25/2019

1. Introduction

This report will be using the 10M movielens dataset which was generated from GroupLens research lab. Edx class provided me code to generate my datasets. Project goal is to develop algorithm using the edx set; for a final test of my algorithm, movie ratings will be predicted in the validation set as if they were unknown. RMSE will be used to evaluate how close my predictions are to the true values in the validation set. Key steps are including: evaluate all the predictors in edx train set; train different algorithms using rating in edx set. Linear regression and regularization will be used to calculate RMSE. For the final test, two best models will be applied on full edx set and calcualte RMSE with validation set. At the end of this report, two models will be recommanded based on lowest RMSE.

2. Methods/Analysis

2.1 load library and data set

```
# Load library
library(tidyverse)
library(caret)
library(data.table)
library(lubridate)
library(ggplot2)
# load data set
# 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")
# 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,]
```

```r
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)
# remove temperary data sets
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

2.1 Create train set and test set from dex, test will be 20% of edx set

```r
set.seed(1, sample.kind="Rounding")
edx_test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.2, list = FALSE)
edx_train <- edx[-edx_test_index,]
edx_temp <- edx[edx_test_index,]
edx_test <- edx_temp %>%
  semi_join(edx_train, by = "movieId")%>%
  semi_join(edx_train, by = "userId")
removed_edx_test <- anti_join(edx_temp, edx_test)
edx_train <- rbind(edx_train, removed_edx_test)
rm(edx_test_index, edx_temp, removed_edx_test)
# take a look of train and test sets
class(edx_train)
```

```
## [1] "data.frame"
```

```r
glimpse(edx_train)
```

```
## Observations: 7,200,089
## Variables: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2...
## $ movieId   <dbl> 185, 316, 329, 355, 364, 377, 420, 539, 588, 589, 61...
## $ rating    <dbl> 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5....
## $ timestamp <int> 838983525, 838983392, 838983392, 838984474, 83898370...
## $ title     <chr> "Net, The (1995)", "Stargate (1994)", "Star Trek: Ge...
## $ genres    <chr> "Action|Crime|Thriller", "Action|Adventure|Sci-Fi", ...
```

```r
# edx_train has 6 vaibles, 7,200,089 observations
class(edx_test)
```

```
## [1] "data.frame"
```

```r
glimpse(edx_test)
```

```
## Observations: 1,799,966
## Variables: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3...
```

```
## $ movieId   <dbl> 122, 292, 356, 362, 370, 466, 520, 594, 260, 376, 53...
## $ rating    <dbl> 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 5.0, 3.0, 3....
## $ timestamp <int> 838985046, 838983421, 838983653, 838984885, 83898459...
## $ title     <chr> "Boomerang (1992)", "Outbreak (1995)", "Forrest Gump...
## $ genres    <chr> "Comedy|Romance", "Action|Drama|Sci-Fi|Thriller", "C...
```

```r
# edx_test has 6 vaibles, 1,799,966 observations
# the column "rating" is outcome, "userId", "movieId", "timestamp",
# "title", and "genres" are 5 preditors.
# take a look of predictor's features
# quantitative preditors, integer or numeric
class(edx_train$userId)
```

```
## [1] "integer"
```

```r
class(edx_train$movieId)
```

```
## [1] "numeric"
```

```r
class(edx_train$timestamp)
```

```
## [1] "integer"
```

```r
# qualitative predictors, charactor
class(edx_train$title)
```

```
## [1] "character"
```

```r
class(edx_train$genres)
```

```
## [1] "character"
```

```r
# Outcome, y, numeric
class(edx_train$rating)
```

```
## [1] "numeric"
```

2.2 Analyze edx train and test set

```r
# take a look train set
summary(edx_train)
```

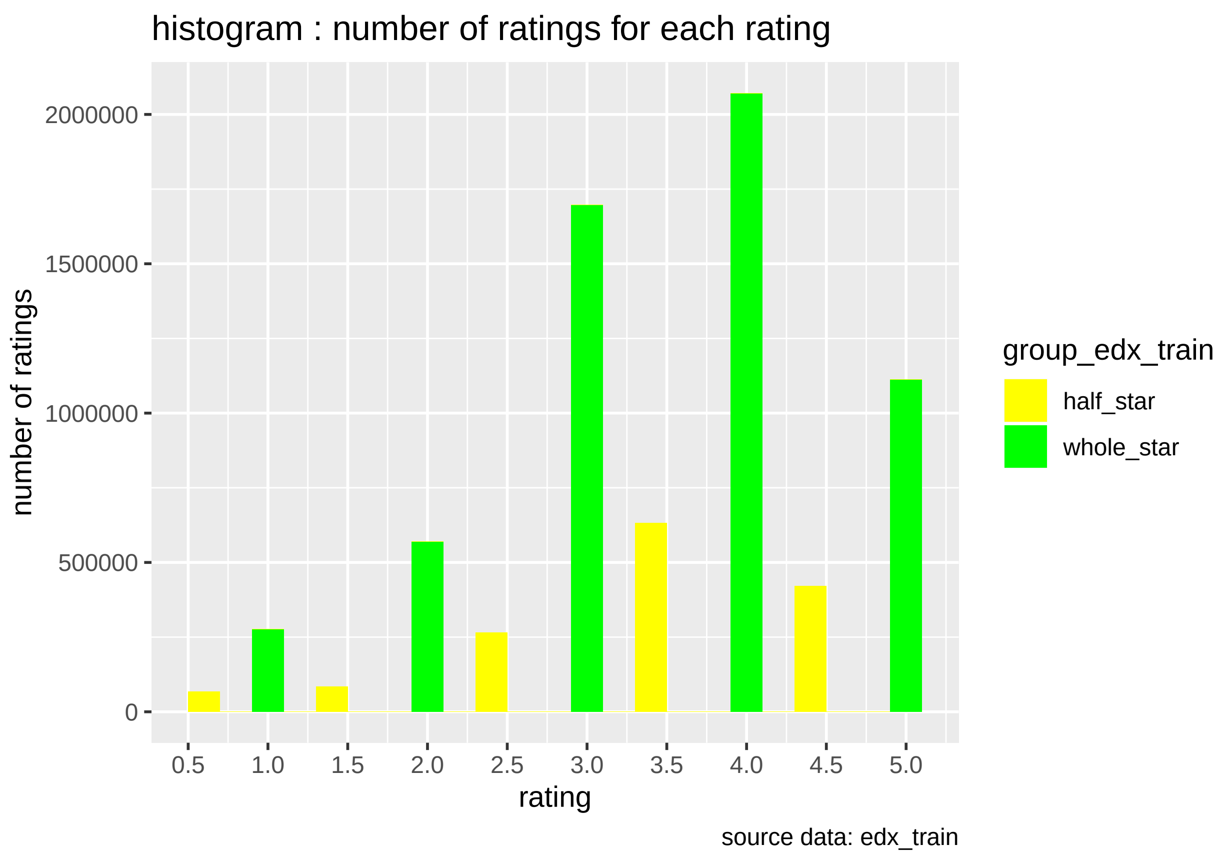
```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18127   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35750   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35873   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
```

```
##  3rd Qu.:53611   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:7200089     Length:7200089
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```
# analyze rating from train set
zero_rating <- edx_train %>% filter(rating == 0)%>%
  summarize(zero = n())
zero_rating
```

```
##   zero
## 1    0
```

```
# zero_rating shows no user gives 0 as rating
group_edx_train <- ifelse(edx_train$rating == 1 |edx_train$rating == 2 |
                            edx_train$rating == 3 |edx_train$rating == 4 |
                            edx_train$rating == 5, "whole_star", "half_star")
explor_ratings_edx_train <- data.frame(edx_train$rating, group_edx_train)
ggplot(explor_ratings_edx_train, aes(x= edx_train$rating, fill = group_edx_train))+
  geom_histogram( binwidth = 0.2)+
  scale_x_continuous(breaks=seq(0, 5, by= 0.5))+
  scale_fill_manual(values = c("half_star"="yellow", "whole_star"="green"))+
  labs(x="rating", y="number of ratings", caption = "source data: edx_train")+
  ggtitle("histogram : number of ratings for each rating")
```

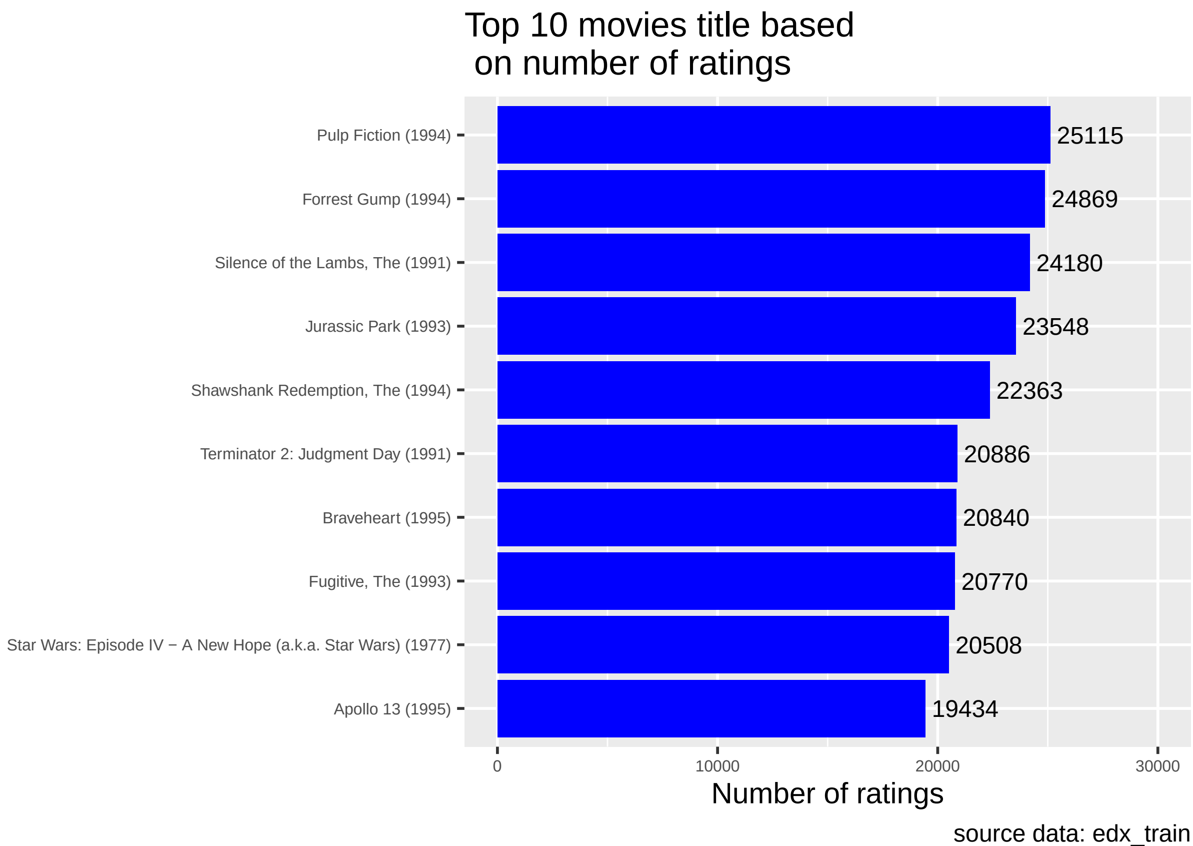



```r
# histogram shows top 5 ratings from most to least are :  4, 3, 5, 3.5 and 2.
# histogram shows that the whole star ratings are more common than half star ratings.
# analyze qualitative predictors: genres, title
# title effect
edx_top_title <- edx_train %>%
  group_by(title) %>%
  summarize(count=n()) %>%
  top_n(10,count) %>%
  arrange(desc(count))
head(edx_top_title, 5)
```

```
## # A tibble: 5 x 2
##   title                              count
##   <chr>                              <int>
## 1 Pulp Fiction (1994)                25115
## 2 Forrest Gump (1994)                24869
## 3 Silence of the Lambs, The (1991) 24180
## 4 Jurassic Park (1993)               23548
## 5 Shawshank Redemption, The (1994) 22363
```

```r
edx_top_title %>% ggplot(aes(x=reorder(title, count), y=count)) +
  geom_bar(stat='identity', fill="blue") + coord_flip(y=c(0, 30000)) +
  labs(x="", y="Number of ratings") +
  geom_text(aes(label= count), hjust=-0.1, size=3) +
  labs(title="Top 10 movies title based \n on number of ratings" ,
```

```
    caption = "source data: edx_train")+
  theme(axis.text = element_text(size = 6))
```

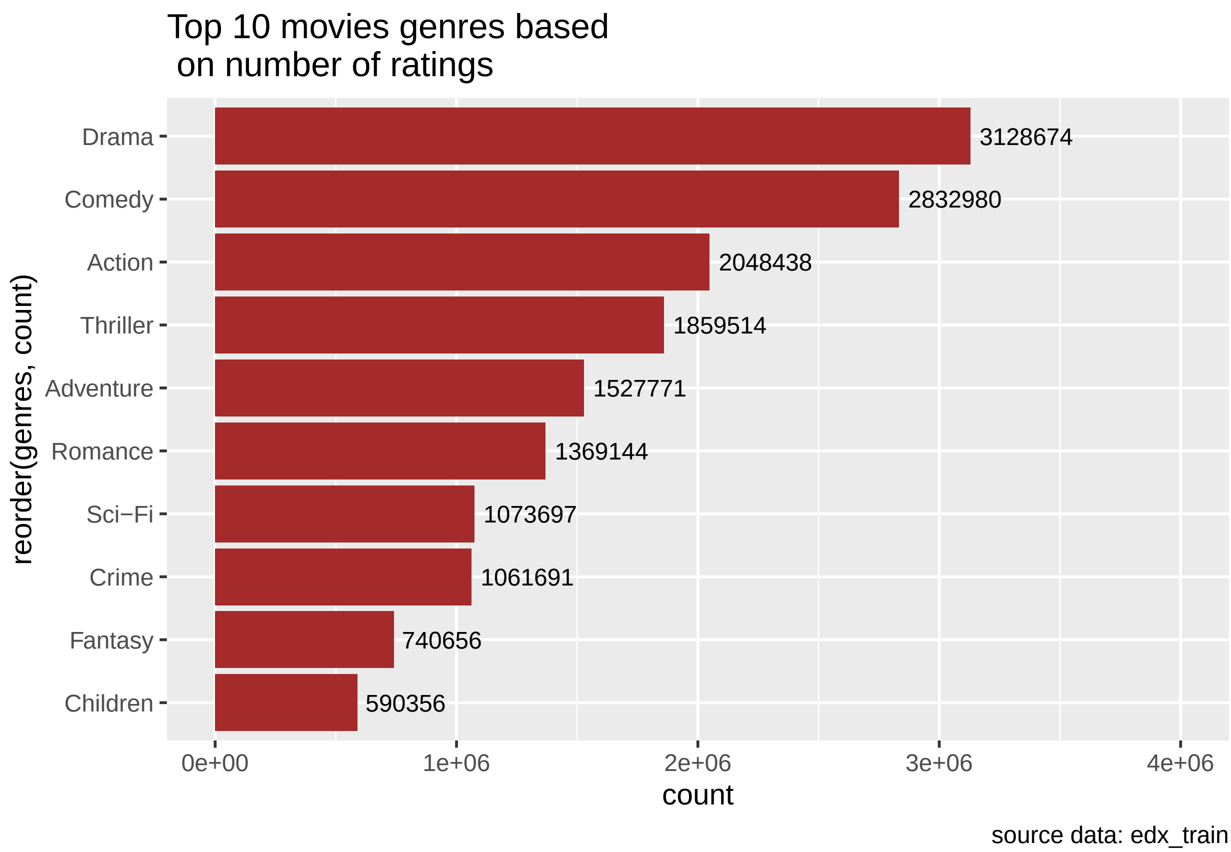



```
# plot shows title effect
# genres effect, separate genres, such as Drama, not Drama/Comedy
edx_top_genr <- edx_train %>% separate_rows(genres, sep = "\\|") %>% group_by(genres) %>%
  summarize(count = n()) %>%
  top_n(10,count)%>%
  arrange(desc(count))
head(edx_top_genr, 5)
```

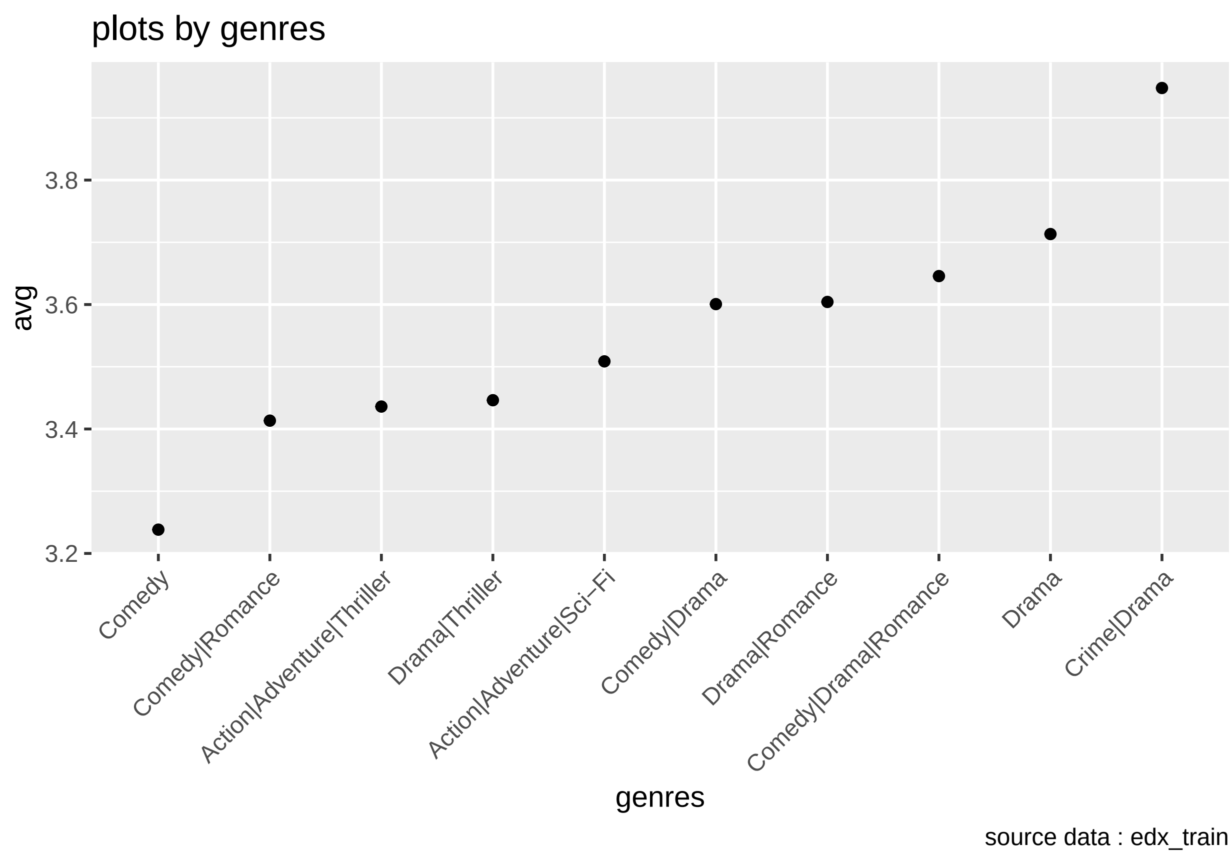
```
## # A tibble: 5 x 2
##   genres      count
##   <chr>       <int>
## 1 Drama     3128674
## 2 Comedy    2832980
## 3 Action    2048438
## 4 Thriller  1859514
## 5 Adventure 1527771
```

```
edx_top_genr %>%
  ggplot(aes(x=reorder(genres, count), y=count)) +
  geom_bar(stat='identity', fill="brown") + coord_flip(y=c(0, 4000000)) +
  geom_text(aes(label= count), hjust=-0.1, size=3) +
```

```r
  labs(title="Top 10 movies genres based \n on number of ratings" ,
       caption = "source data: edx_train")
```



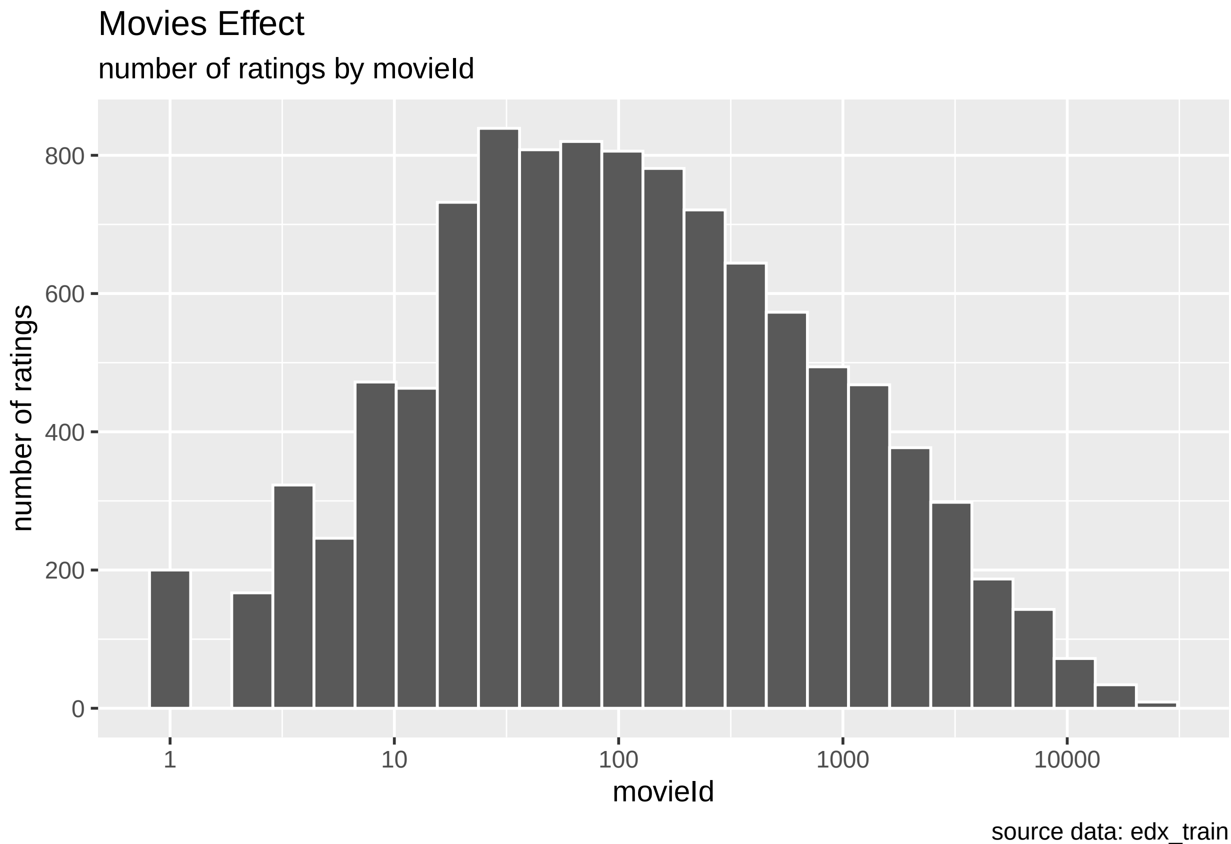

```r
# plot shows genres effect
# futher investigate genres effect, all combinations, such as Drama and Drama/Comedy
edx_train %>% group_by(genres) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n >= 100000) %>%
  mutate(genres = reorder(genres, avg)) %>%
  ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point()+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))+
  labs(title = "plots by genres" , caption = "source data : edx_train")
```



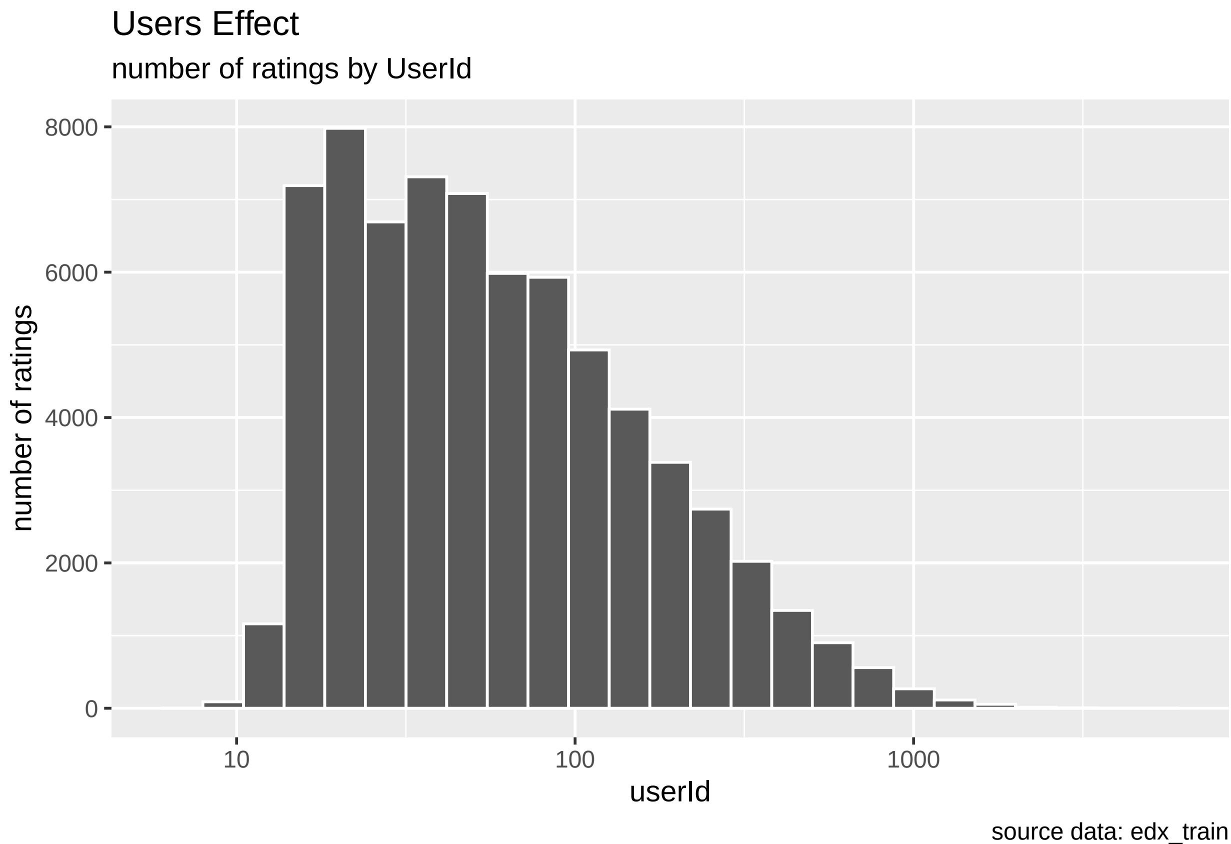

```r
# plot shows strong evidence of a genre effect
# quantitative features: UserId, movieId, timestamp
# take a look of userId, movieId, and title.
edx_train %>%
  summarize(n_user = n_distinct(userId),
            n_movies = n_distinct(movieId),
            n_title = n_distinct(title))
```

```
##   n_user n_movies n_title
## 1  69878    10677   10676
```

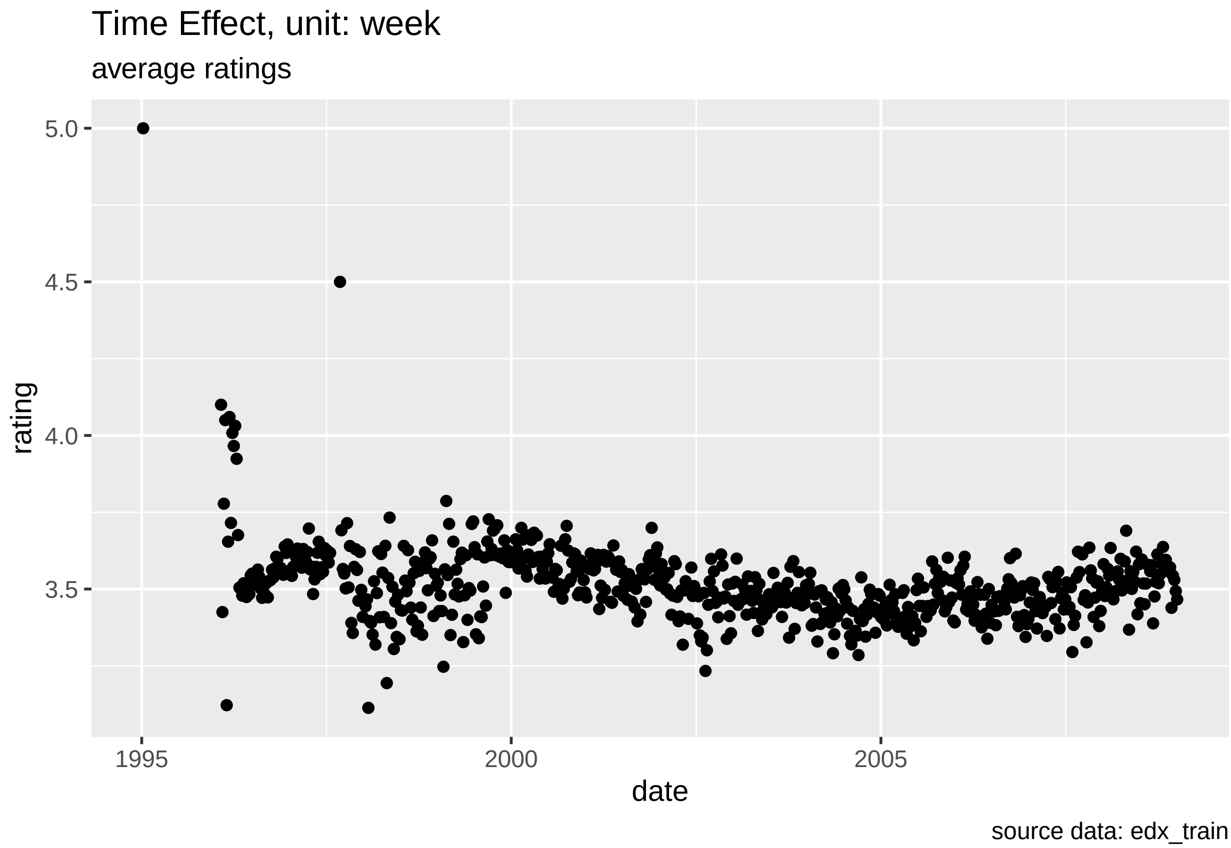
```r
# title is not unique, movieId is unique. userId is not unique.
# histogram of number of ratings by movieId
edx_train %>%
  count(movieId) %>%
  ggplot(aes(n))+
  geom_histogram(bins = 25, color = "white")+
  scale_x_log10()+
  ggtitle("Movies Effect")+
  labs(subtitle = "number of ratings by movieId",
       x = "movieId",
       y = "number of ratings",
       caption = "source data: edx_train")
```




```
# plot shows strong movieId effect
# histogram of number of ratings by userId
edx_train %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram( bins=25, color = "white") +
  scale_x_log10() +
  ggtitle("Users Effect") +
  labs(subtitle ="number of ratings by UserId",
       x="userId" ,
       y="number of ratings",
       caption = "source data: edx_train")
```




```
# plot shows strong userId effect
# plot of timestamp
edx_train %>% mutate(date = round_date(as_datetime(timestamp),unit = "week")) %>%
  group_by(date) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(date, rating))+
  geom_point()+
  ggtitle("Time Effect, unit: week")+
  labs(subtitle = "average ratings",
       caption = "source data: edx_train")
```




```
# plot shows timestamp effect, but not strong.
# Overall summary of these 5 predictors, movieId, userId, title, and genres have strong
# effect, timestamp has weak effect.
# Both title and movieId have strong effect, because title is not unique, movieId is
# a better predictor for movie effect.
```

2.3 method and RMSE by usging edx train and test set

```
# 2.3.1. Just average, no predictors effect
# 2.3.2. Regression Models
#  movie effect
#  movie + user effect
#  movie + user + time effect
#  movie + user + genres effect
# 2.3.3. Regularization + movie + user effect
# Define RMSE function:
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))}
# calculate RMSE from edx_train and edx_test
mu <- mean(edx_train$rating)
# Just average, no predictors effect
model_avgs_rmse <- RMSE(edx_test$rating, mu)
model_avgs_rmse
```

```
## [1] 1.059904
```

```r
options(pillar.sigfig = 7)
rmse_results <- tibble(method = "just average", RMSE = model_avgs_rmse)
# movie effect, use movieId, not title
edx_movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating -mu))
edx_test_temp <- edx_test %>%
  left_join(edx_movie_avgs, by='movieId')%>%
  .$b_i
predicted_ratings_bi <- mu + edx_test_temp
rm(edx_test_temp)
model_1_rmse <- RMSE(predicted_ratings_bi, edx_test$rating)
model_1_rmse
```

```
## [1] 0.9437429
```

```r
rmse_results <- bind_rows(rmse_results,
                          tibble(method="movie effect",
                                 RMSE = model_1_rmse))
# movie + user effect
edx_user_avgs <- edx_train %>%
  left_join(edx_movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
edx_test_temp <- edx_test %>%
  left_join(edx_movie_avgs, by='movieId') %>%
  left_join(edx_user_avgs, by='userId') %>%
  .$b_u
predicted_ratings_bi_bu <- predicted_ratings_bi + edx_test_temp
rm(edx_test_temp)
model_2_rmse <- RMSE(predicted_ratings_bi_bu, edx_test$rating)
model_2_rmse
```

```
## [1] 0.8659319
```

```r
rmse_results <- bind_rows(rmse_results,
                          tibble(method="movie + user effect",
                                 RMSE = model_2_rmse))
# movie + user + genres effect
edx_genres <- edx_train %>%
  left_join(edx_movie_avgs, by='movieId') %>%
  left_join(edx_user_avgs, by='userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating-mu-b_i-b_u))
edx_test_temp <- edx_test %>%
  left_join(edx_movie_avgs, by='movieId') %>%
  left_join(edx_user_avgs, by='userId') %>%
  left_join(edx_genres, by = 'genres')%>%
  .$b_g
predicted_ratings_bi_bu_bg <- predicted_ratings_bi_bu + edx_test_temp
```

```r
rm(edx_test_temp)
model_3_rmse <- RMSE(predicted_ratings_bi_bu_bg, edx_test$rating)
model_3_rmse
```

```
## [1] 0.8655941
```

```r
rmse_results <- bind_rows(rmse_results,
                          tibble(method="movie + user + genres effect",
                                 RMSE = model_3_rmse))
# genres doesn't show strong impact on RMSE
# I learned timestamp effect is not strong, but I would like to see the timestamp
# impact on RMSE
edx_time <- edx_train %>%
  left_join(edx_movie_avgs, by='movieId') %>%
  left_join(edx_user_avgs, by='userId') %>%
  left_join(edx_genres, by = 'genres')%>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  group_by(date) %>%
  summarize(b_t = mean(rating - mu - b_i - b_u - b_g))
edx_test_temp <- edx_test %>%
  left_join(edx_movie_avgs, by='movieId') %>%
  left_join(edx_user_avgs, by='userId') %>%
  left_join(edx_genres, by = 'genres')%>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  left_join(edx_time, by = 'date') %>%
  .$b_t
predicted_ratings_bi_bu_bg_bt <- predicted_ratings_bi_bu_bg + edx_test_temp
rm(edx_test_temp)
model_4_rmse <- RMSE(predicted_ratings_bi_bu_bg_bt, edx_test$rating)
model_4_rmse
```

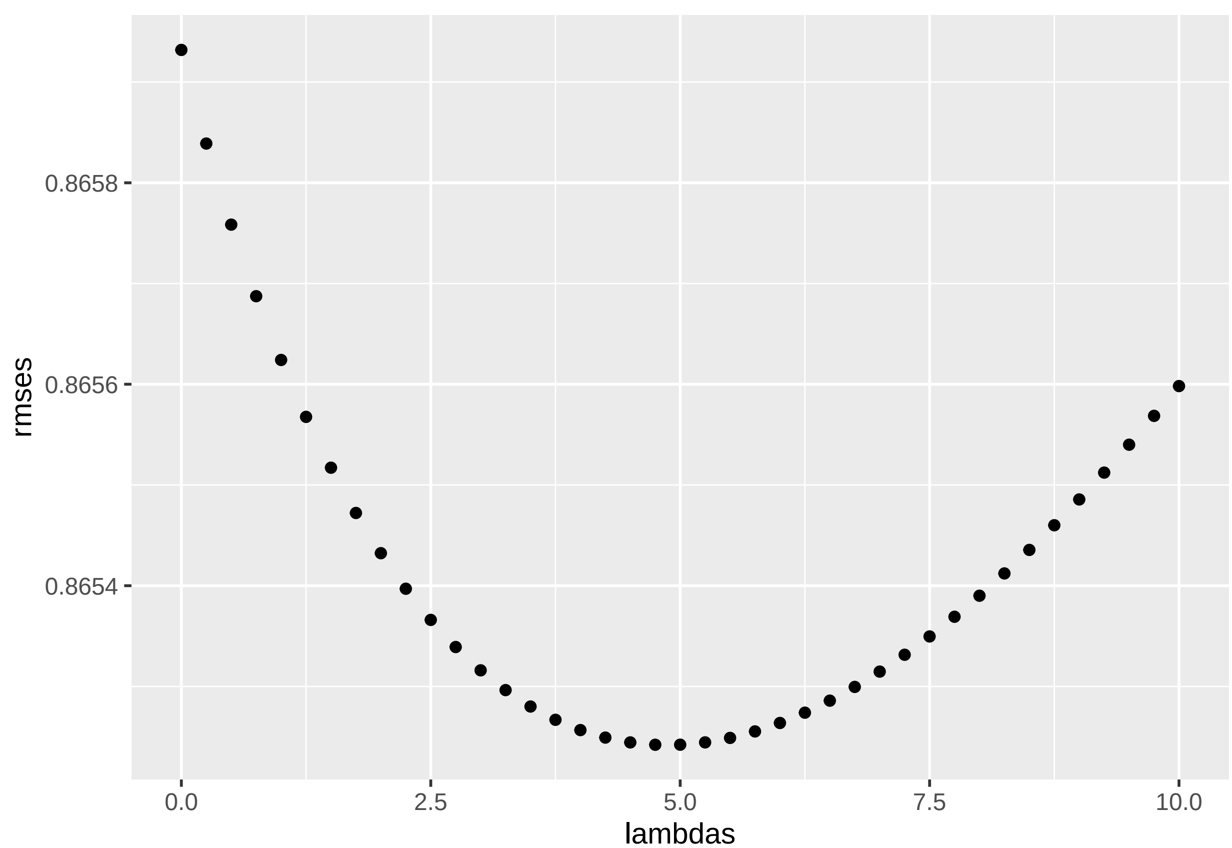
```
## [1] 0.8654875
```

```r
rmse_results <- bind_rows(rmse_results,
                          tibble(method="movie + user + genres + time effect",
                                 RMSE = model_4_rmse))
rmse_results
```

```
## # A tibble: 5 x 2
##   method                                  RMSE
##   <chr>                                  <dbl>
## 1 just average                        1.059904
## 2 movie effect                        0.9437429
## 3 movie + user effect                 0.8659319
## 4 movie + user + genres effect        0.8655941
## 5 movie + user + genres + time effect 0.8654875
```

```r
# rmse results shows different RMSE results, movieId and userId have strong impact
# use regularization to optimize model with movie and user
# use cross-validatoin to optimize lambda
lambdas <- seq(0, 10, 0.25)
```

```
rmses <- sapply(lambdas, function(l){
  mu_reg <- mean(edx_train$rating)
  b_i_reg <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i_reg = sum(rating - mu_reg)/(n()+l))
  b_u_reg <- edx_train %>%
    left_join(b_i_reg, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u_reg = sum(rating - b_i_reg - mu_reg)/(n()+l))
  predicted_ratings_b_i_u <-
    edx_test %>%
    left_join(b_i_reg, by = "movieId") %>%
    left_join(b_u_reg, by = "userId") %>%
    mutate(pred = mu_reg + b_i_reg + b_u_reg) %>%
    .$pred
  return(RMSE(edx_test$rating,predicted_ratings_b_i_u))
})
qplot(lambdas, rmses)
```




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

```
## [1] 4.75
```

```r
model_5_rmse <- min(rmses)
model_5_rmse
```

```
## [1] 0.8652421
```

```r
rmse_results <- bind_rows(rmse_results,
                          tibble(method="movie + user + regularization",
                                 RMSE = model_5_rmse))
rmse_results
```

```
## # A tibble: 6 x 2
##   method                                  RMSE
##   <chr>                                  <dbl>
## 1 just average                        1.059904
## 2 movie effect                        0.9437429
## 3 movie + user effect                 0.8659319
## 4 movie + user + genres effect        0.8655941
## 5 movie + user + genres + time effect 0.8654875
## 6 movie + user + regularization       0.8652421
```

```r
# the last two models show minium RMSE in edx set
```

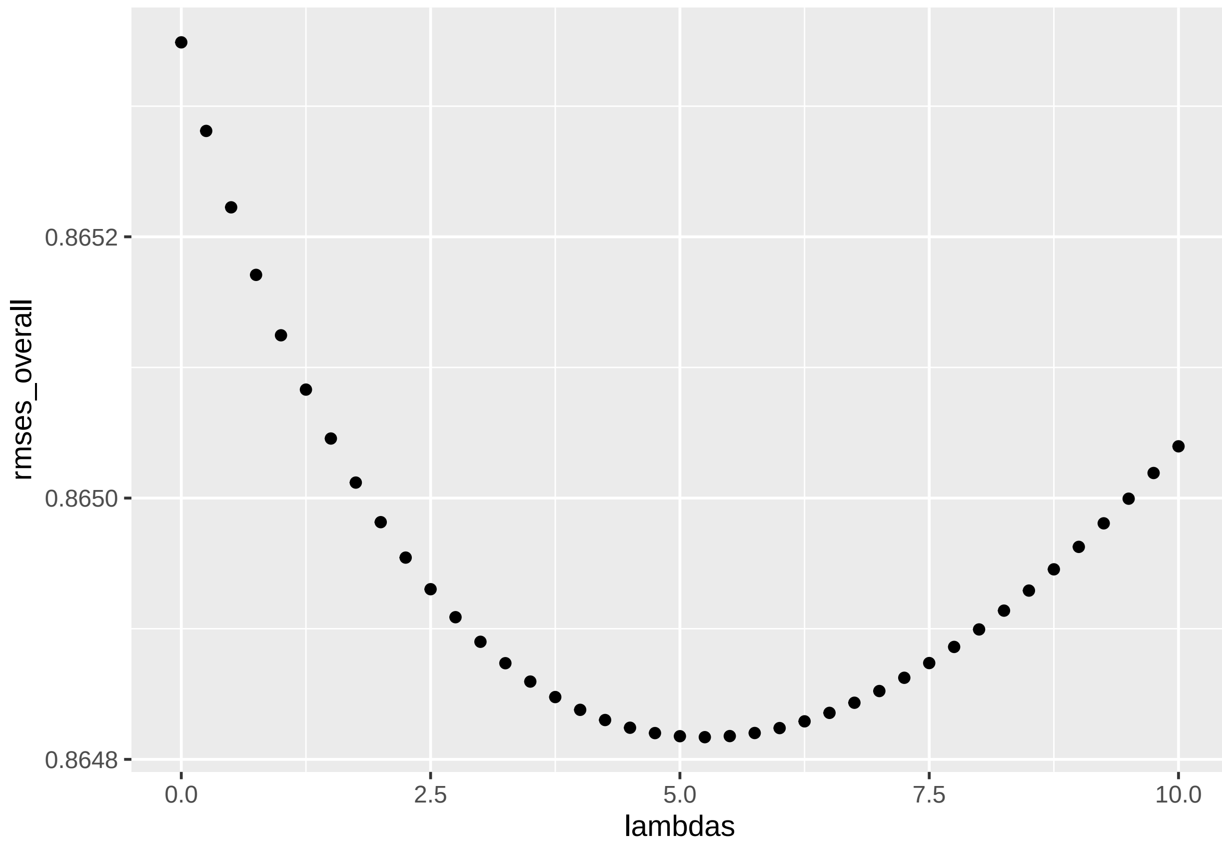
3. Results

```r
# apply the last two models: Linear regression with movie + user + genres + time effect
# and regularization + movie + user effect into validation set for RMSE.
mu_edx <- mean(edx$rating)
movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating -mu_edx))
user_avgs <- edx %>%
  left_join(movie_avgs, by = "movieId")%>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_edx - b_i))
genres_effect <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating-mu_edx-b_i-b_u))
time_effect <- edx %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  left_join(genres_effect, by ='genres')%>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  group_by(date)%>%
  summarize(b_t = mean(rating - mu_edx - b_i - b_u - b_g))
predicted_ratings_b_iugt <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
```

```
  left_join(genres_effect, by ='genres')%>%
  mutate(date = round_date(as_datetime(timestamp), unit = "week")) %>%
  left_join(time_effect, by ='date')%>%
  mutate(pred = mu_edx+b_i+b_u+b_g+b_t)%>%
  .$pred
final_model_1 <- RMSE(predicted_ratings_b_iugt, validation$rating)
final_model_1
```

```
## [1] 0.8648392
```

```
final_rmses <- tibble(model = "movie + user + genres + time effect",
                      RMSE = final_model_1)
# apply regularization with movie and user effect
lambdas <- seq(0, 10, 0.25)
rmses_overall <- sapply(lambdas, function(l){
  mu_reg <- mean(edx$rating)
  b_i_reg <- edx %>%
    group_by(movieId) %>%
    summarize(b_i_reg = sum(rating - mu_reg)/(n()+l))
  b_u_reg <- edx %>%
    left_join(b_i_reg, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u_reg = sum(rating - b_i_reg - mu_reg)/(n()+l))
  predicted_ratings_b_i_u <-
    validation %>%
    left_join(b_i_reg, by = "movieId") %>%
    left_join(b_u_reg, by = "userId") %>%
    mutate(pred = mu_reg + b_i_reg + b_u_reg) %>%
    .$pred
  return(RMSE(validation$rating,predicted_ratings_b_i_u))
})
qplot(lambdas, rmses_overall)
```




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

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

```
final_model_2<- min(rmses_overall)
final_model_2
```

```
## [1] 0.864817
```

```
final_rmses <- bind_rows(final_rmses,
                         tibble(model ="movie + user + regularization",
                                RMSE = final_model_2))
final_rmses
```

```
## # A tibble: 2 x 2
##   model                                   RMSE
##   <chr>                                  <dbl>
## 1 movie + user + genres + time effect 0.8648392
## 2 movie + user + regularization       0.8648170
```

4. Conclusion

I tried the potential best algorithm to predict movie ratings for the 10M version of the MovieLens data. By using the provided edx and validation set from our class, I trained different linear regression modle plus regularization. RMSE was used to evaluate model performance. As my conclusion, the linear regression model with movie, user, genres, and time effects gave me RMSE 0.8648392, and the regularization model with movie and user effects gave me RMSE 0.8648170. Both RMSE are less than 0.8649. Future work will be using recommenderlab package, Matrix Factorization method, Ensemble method to further investigate lower RMSE.