Movielens Project

Cheol Min Lee

7/16/2021

Introduction

This project is to develop a movie recomendation system using the MoveLens dataset. Recommendation system is one of the most succesful and widespread application of machine learning technologies in business. For the movielens project, I used the 10M version of the MovieLens dataset.The goal of this project is to build a movie recommendation system using machine learning. The first step will be to look at the structure of the data, visualize it and then progressively build a model that will reach the targeted accuracy.

Data Loading

The code is provided in the EDx capstone project module

```
# Create test and validation sets
# Create edx set, validation set, and submission file
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(stringr)) install.packages("stringr", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(simtimer)) install.packages("simtimer", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(tidyr)) install.packages("tidyr", repos = "http://cran.us.r-project.org")
if(!require(corrplot)) install.packages("corrplot", repos = "http://cran.us.r-project.org")
if(!require(RColorBrewer)) install.packages("RColorBrewer", repos = "http://cran.us.r-project.org")
if(!require(hexbin)) install.packages("hexbin", repos = "http://cran.us.r-project.org")
if(!require(Metrics)) install.packages("Metrics", repos = "http://cran.us.r-project.org")
if(!require(gridExtra)) install.packages("gridExtra", repos = "http://cran.us.r-project.org")

library(dplyr)
library(stringr)
library(caret)
library(simtimer)
library(lubridate)
library(ggplot2)
library(tidyr)
library(corrplot)
library(RColorBrewer)
library(hexbin)
library(Metrics)
library(gridExtra)

# MovieLens 10M dataset:
```

```r
# 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 <- read.table(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")

# Validation set will be 10% of MovieLens data
set.seed(2021)
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)
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 3      1     231      5 838983392          Dumb & Dumber (1994)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
## 8      1     356      5 838983653           Forrest Gump (1994)
##                          genres
## 3                        Comedy
## 4  Action|Drama|Sci-Fi|Thriller
## 5       Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
## 8      Comedy|Drama|Romance|War
```

```r
# How many distinct movie, users and genres in movielens data
n_distinct(edx$movieId)
```

```
## [1] 10677
```

```r
n_distinct(edx$genres)
```

```
## [1] 797
```

```
n_distinct(edx$userId)
```

```
## [1] 69878
```

Extract the premier date and calculate the age of the move

```
# Change from Timestamp to year
edx <- mutate(edx, year_rated = year(as_datetime(timestamp)))
head(edx)
```

```
##   userId movieId rating timestamp                          title
## 3      1     231      5 838983392          Dumb & Dumber (1994)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
## 8      1     356      5 838983653           Forrest Gump (1994)
##                          genres year_rated
## 3                        Comedy       1996
## 4  Action|Drama|Sci-Fi|Thriller       1996
## 5       Action|Adventure|Sci-Fi       1996
## 6 Action|Adventure|Drama|Sci-Fi       1996
## 7       Children|Comedy|Fantasy       1996
## 8      Comedy|Drama|Romance|War       1996
```

```
# Extract the premier date
premier <- stringi::stri_extract(edx$title, regex = "(\\d{4})", comments = TRUE ) %>% as.numeric()

# Add the premier date
edx_with_title_dates <- edx %>% mutate(premier_date = premier)
head(edx_with_title_dates)
```

```
##   userId movieId rating timestamp                          title
## 3      1     231      5 838983392          Dumb & Dumber (1994)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
## 8      1     356      5 838983653           Forrest Gump (1994)
##                          genres year_rated premier_date
## 3                        Comedy       1996         1994
## 4  Action|Drama|Sci-Fi|Thriller       1996         1995
## 5       Action|Adventure|Sci-Fi       1996         1994
## 6 Action|Adventure|Drama|Sci-Fi       1996         1994
## 7       Children|Comedy|Fantasy       1996         1994
## 8      Comedy|Drama|Romance|War       1996         1994
```

```
# Drop the timestamp
edx_with_title_dates <- edx_with_title_dates %>% select(-timestamp)
head(edx_with_title_dates)
```

```
##   userId movieId rating                          title
## 3      1     231      5          Dumb & Dumber (1994)
## 4      1     292      5               Outbreak (1995)
## 5      1     316      5               Stargate (1994)
## 6      1     329      5 Star Trek: Generations (1994)
## 7      1     355      5       Flintstones, The (1994)
## 8      1     356      5           Forrest Gump (1994)
##                          genres year_rated premier_date
## 3                        Comedy       1996         1994
## 4  Action|Drama|Sci-Fi|Thriller       1996         1995
## 5       Action|Adventure|Sci-Fi       1996         1994
## 6 Action|Adventure|Drama|Sci-Fi       1996         1994
## 7       Children|Comedy|Fantasy       1996         1994
## 8      Comedy|Drama|Romance|War       1996         1994
```

```
# Check the dates
edx_with_title_dates %>% filter(premier_date > 2021) %>% group_by(movieId, title, premier_date) %>% summ
```

```
## `summarise()` has grouped output by 'movieId', 'title'. You can override using the `.groups` argumen
```

```
## # A tibble: 6 x 4
## # Groups:   movieId, title [6]
##   movieId title                                          premier_date     n
##     <dbl> <chr>                                                 <dbl> <int>
## 1     671 Mystery Science Theater 3000: The Movie (1996)         3000  3278
## 2    2308 Detroit 9000 (1973)                                    9000    18
## 3    4159 3000 Miles to Graceland (2001)                         3000   720
## 4    5310 Transylvania 6-5000 (1985)                             5000   193
## 5    8864 Mr. 3000 (2004)                                        3000   145
## 6   27266 2046 (2004)                                            2046   428
```

```
edx_with_title_dates[edx_with_title_dates$movieId == "671", "premier_date"] <- 1996
edx_with_title_dates[edx_with_title_dates$movieId == "2308", "premier_date"] <- 1973
edx_with_title_dates[edx_with_title_dates$movieId == "4159", "premier_date"] <- 2001
edx_with_title_dates[edx_with_title_dates$movieId == "5310", "premier_date"] <- 1985
edx_with_title_dates[edx_with_title_dates$movieId == "8864", "premier_date"] <- 2004
edx_with_title_dates[edx_with_title_dates$movieId == "27266", "premier_date"] <- 2004

edx_with_title_dates %>% filter(premier_date < 1900) %>% group_by(movieId, title, premier_date) %>% summ
```

```
## `summarise()` has grouped output by 'movieId', 'title'. You can override using the `.groups` argumen
```

```
## # A tibble: 8 x 4
## # Groups:   movieId, title [8]
##   movieId title                                                premier_date     n
##     <dbl> <chr>                                                       <dbl> <int>
## 1    1422 Murder at 1600 (1997)                                       1600  1591
## 2    4311 Bloody Angels (1732 HÃ¸tten: Marerittet Har et Pos~         1732     9
## 3    5472 1776 (1972)                                                 1776   187
## 4    6290 House of 1000 Corpses (2003)                                1000   371
## 5    6645 THX 1138 (1971)                                             1138   489
## 6    8198 1000 Eyes of Dr. Mabuse, The (Tausend Augen des Dr~         1000    28
## 7    8905 1492: Conquest of Paradise (1992)                           1492   136
## 8   53953 1408 (2007)                                                 1408   475
```

```
edx_with_title_dates[edx_with_title_dates$movieId == "1422", "premier_date"] <- 1997
edx_with_title_dates[edx_with_title_dates$movieId == "4311", "premier_date"] <- 1998
edx_with_title_dates[edx_with_title_dates$movieId == "5472", "premier_date"] <- 1972
edx_with_title_dates[edx_with_title_dates$movieId == "6290", "premier_date"] <- 2003
edx_with_title_dates[edx_with_title_dates$movieId == "6645", "premier_date"] <- 1971
edx_with_title_dates[edx_with_title_dates$movieId == "8198", "premier_date"] <- 1960
edx_with_title_dates[edx_with_title_dates$movieId == "8905", "premier_date"] <- 1992
edx_with_title_dates[edx_with_title_dates$movieId == "53953", "premier_date"] <- 2007

# Calculate the age of a movie
edx_with_title_dates <- edx_with_title_dates %>% mutate(age_of_movie = 2021 - premier_date,
                                                        rating_date_range = year_rated - premier_date)
head(edx_with_title_dates)
```

```
##   userId movieId rating                         title
## 3      1     231      5          Dumb & Dumber (1994)
## 4      1     292      5               Outbreak (1995)
## 5      1     316      5               Stargate (1994)
## 6      1     329      5 Star Trek: Generations (1994)
## 7      1     355      5       Flintstones, The (1994)
## 8      1     356      5           Forrest Gump (1994)
##                          genres year_rated premier_date age_of_movie
## 3                        Comedy       1996         1994           27
## 4  Action|Drama|Sci-Fi|Thriller       1996         1995           26
## 5       Action|Adventure|Sci-Fi       1996         1994           27
## 6 Action|Adventure|Drama|Sci-Fi       1996         1994           27
## 7       Children|Comedy|Fantasy       1996         1994           27
## 8      Comedy|Drama|Romance|War       1996         1994           27
##   rating_date_range
## 3                 2
## 4                 1
## 5                 2
## 6                 2
## 7                 2
## 8                 2
```

Make graphes based on Movielens data

```
# Make graphes based on Movielens data
Plot1<- edx %>% group_by(movieId) %>% summarize(n = n()) %>%
  ggplot(aes(n)) + geom_histogram(fill = "cadetblue2", color = "grey20", bins = 20) +
  labs(y = "Number of Movie", x = "Rating")+
  scale_x_log10()

# Make Distribution of Users
Plot2<- edx %>% group_by(userId) %>% summarize(n = n()) %>%
  ggplot(aes(n)) + geom_histogram(fill = "71b7begreen", color = "grey20", bins = 20) +
  labs(y = "Number of User", x = "Rating")+
  scale_x_log10()

# Make Figure 1 with Plot1 and Plot2
grid.arrange(Plot1, Plot2, nrow=1, ncol=2)
```

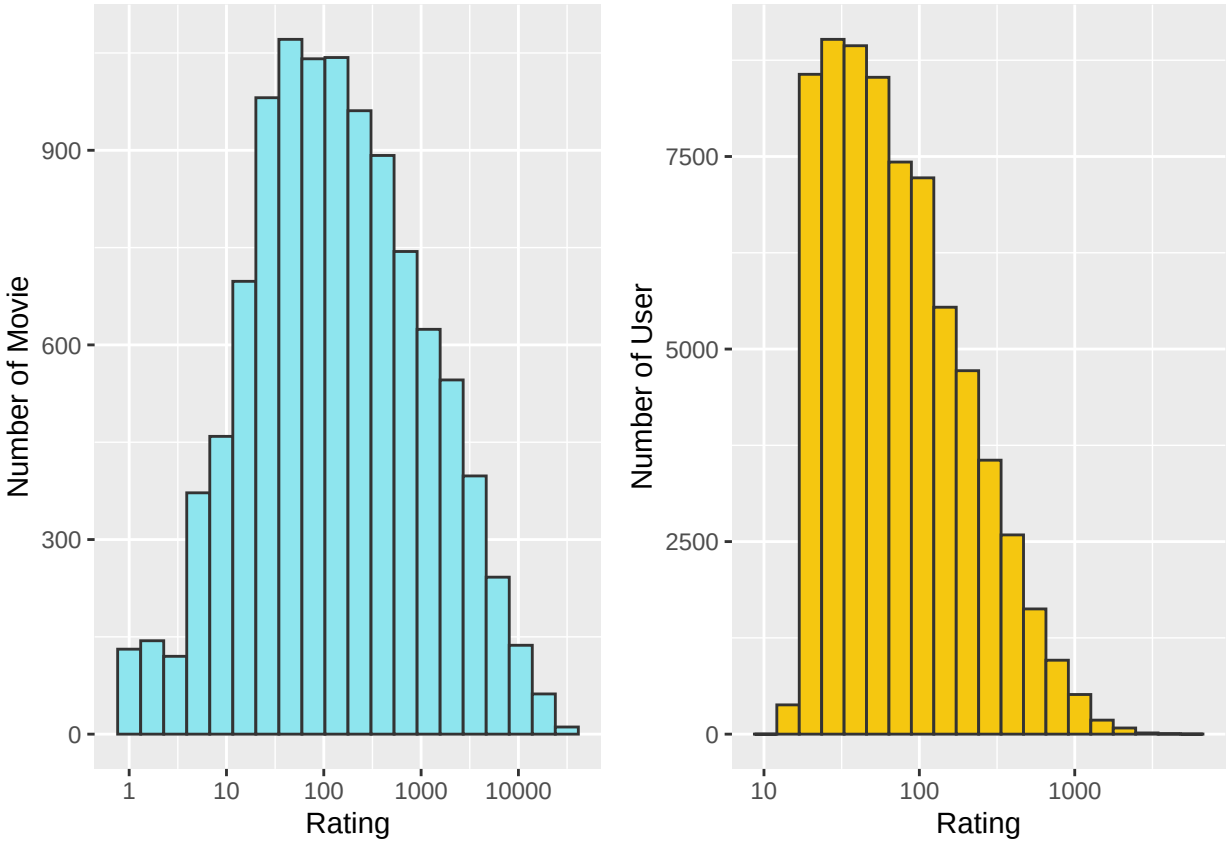


Figure 1. Distribution of movie rating and distribution of user rating in Movielens data.

Calculate movie rating average, user rating average, average rating by age of movie, average rating by year

```
movie_avgs <- edx_with_title_dates %>% group_by(movieId) %>% summarize(avg_movie_rating = mean(rating))
user_avgs <- edx_with_title_dates %>% group_by(userId) %>% summarize(avg_user_rating = mean(rating))
year_avgs <- edx_with_title_dates%>% group_by(year_rated) %>% summarize(avg_rating_by_year = mean(ratin
age_avgs <- edx_with_title_dates %>% group_by(age_of_movie) %>% summarize(avg_rating_by_age = mean(rati
head(age_avgs)
```

```
## # A tibble: 6 x 2
##   age_of_movie avg_rating_by_age
##          <dbl>             <dbl>
## 1           11              3.36
## 2           13              3.46
## 3           14              3.53
## 4           15              3.53
## 5           16              3.48
## 6           17              3.53
```

```
head(user_avgs)
```

```
## # A tibble: 6 x 2
##   userId avg_user_rating
```

```
##    <int>           <dbl>
## 1      1            5
## 2      2            3.22
## 3      3            4.04
## 4      4            4.03
## 5      5            3.85
## 6      6            3.92
```

```
# Age of movie vs average movie rating
Plot3 <- age_avgs %>%
  ggplot(aes(age_of_movie, avg_rating_by_age)) +
  geom_point(alpha = 1, colour = "#99CC00") +
  ggtitle("a. Age of a Movie vs Average Movie Rating")

# UserId vs average movie rating
Plot4 <- user_avgs %>%
  ggplot(aes(userId, avg_user_rating)) +
  geom_point(alpha = 1/10, colour = "#FFCC33") +
  ggtitle("b. User vs Average User Rating")


# Test the linear models
summary(lm(avg_rating_by_age ~ age_of_movie, data = age_avgs))
```

```
##
## Call:
## lm(formula = avg_rating_by_age ~ age_of_movie, data = age_avgs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.63324 -0.10689  0.00344  0.12964  0.28841
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.4682546  0.0428560  80.928  < 2e-16 ***
## age_of_movie 0.0041201  0.0006504   6.334 8.14e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1785 on 94 degrees of freedom
## Multiple R-squared:  0.2992, Adjusted R-squared:  0.2917
## F-statistic: 40.12 on 1 and 94 DF,  p-value: 8.139e-09
```

```
summary(lm(avg_user_rating~ userId, data = user_avgs))
```

```
##
## Call:
## lm(formula = avg_user_rating ~ userId, data = user_avgs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.11505 -0.25651  0.02145  0.28871  1.38802
##
```

```
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 3.612e+00  3.265e-03 1106.164   <2e-16 ***
## userId      4.878e-08  7.907e-08    0.617    0.537
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4309 on 69876 degrees of freedom
## Multiple R-squared:  5.447e-06,  Adjusted R-squared:  -8.864e-06
## F-statistic: 0.3806 on 1 and 69876 DF,  p-value: 0.5373
```

```
# Make Figure 2 with Plot3 and Plot4
grid.arrange(Plot3, Plot4, nrow=1, ncol=2)
```

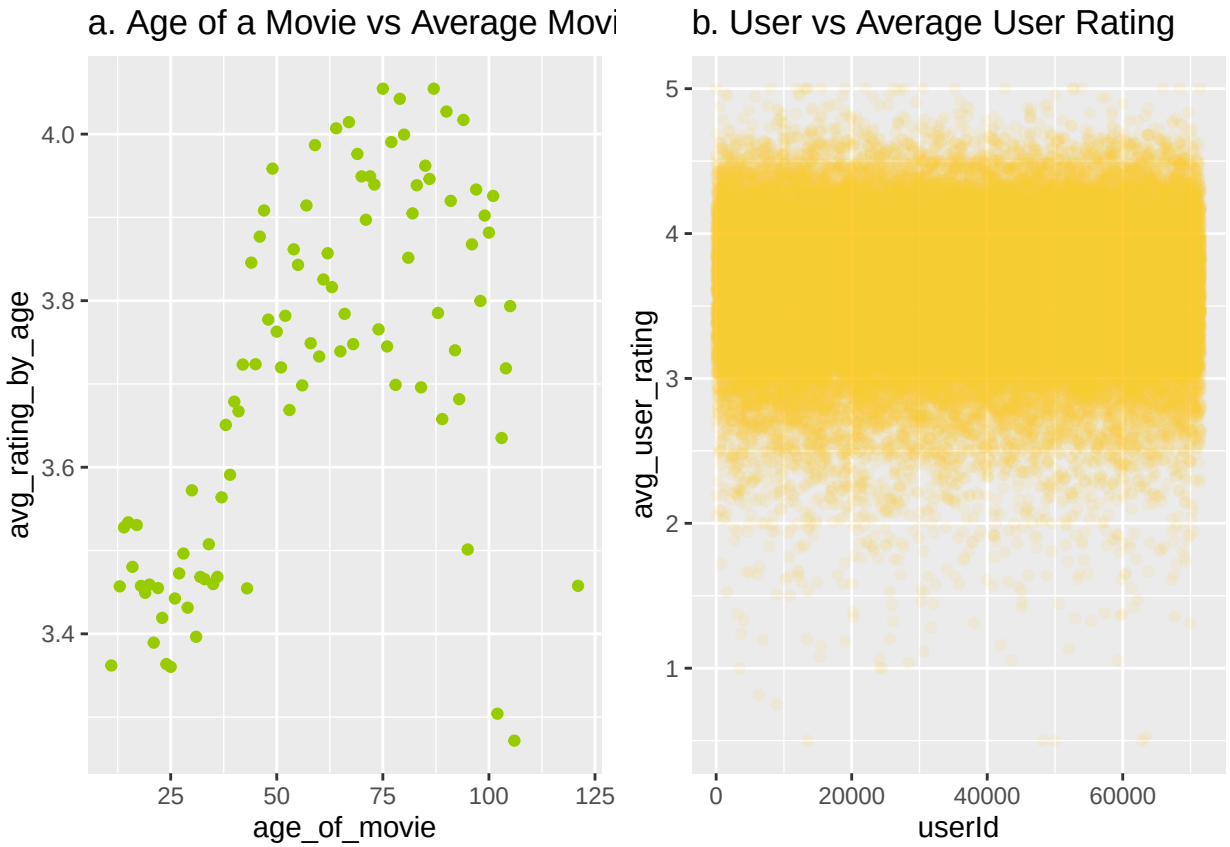


Figure 2. Relationship between age of movie (a) and average rating by age and user and average user rating (b).

Relationship between age of movie and average rating by age

```
# Movies less than 60 years old
age_of_movie_less_than60 <- age_avgs %>% filter(age_of_movie <60)
Plot5 <- age_of_movie_less_than60 %>%
  ggplot(aes(age_of_movie, avg_rating_by_age)) +
  geom_point(alpha = 1, colour = "#FF6633", size=5) +
  ggtitle("a. Less than 60 years old")
```

```r
# Movies between 30 and 60 years old
age_between30_and_60 <- age_avgs %>% filter((age_of_movie > 30) & (age_of_movie < 60))
Plot6 <- age_between30_and_60 %>%
  ggplot(aes(age_of_movie, avg_rating_by_age)) +
  geom_point(alpha = 1, colour = "#CC0033", size=5) +
  ggtitle("b. Between 30 and 60 years old")


# Movies between 10 and 50 years old
age_between10_and_50 <- age_avgs %>% filter((age_of_movie > 10) & (age_of_movie < 50))
Plot7 <- age_between10_and_50 %>%
  ggplot(aes(age_of_movie, avg_rating_by_age)) +
  geom_point(alpha = 1, colour = "#3399FF", size=5) +
  ggtitle('c. Between 10 and 50 years old')


# Movies movie between 20 and 40 years old
age_between20_and_40 <- age_avgs %>% filter((age_of_movie > 20) & (age_of_movie < 40))
Plot8 <- age_between20_and_40 %>%
  ggplot(aes(age_of_movie, avg_rating_by_age)) +
  geom_point(alpha = 1, colour = "#CC00CC", size=5) +
  ggtitle('d. Between 10 and 40 years old')


summary(lm(avg_rating_by_age ~ age_of_movie, data = age_of_movie_less_than60)) #The R-squared is 0.6812
```

```
##
## Call:
## lm(formula = avg_rating_by_age ~ age_of_movie, data = age_of_movie_less_than60)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.225810 -0.080731  0.008078  0.056299  0.214538
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.22455    0.04076  79.103  < 2e-16 ***
## age_of_movie  0.01060    0.00107   9.909 5.44e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.103 on 46 degrees of freedom
## Multiple R-squared:  0.681,  Adjusted R-squared:  0.674
## F-statistic: 98.18 on 1 and 46 DF,  p-value: 5.444e-13
```

```r
summary(lm(avg_rating_by_age ~ age_of_movie, data = age_between30_and_60)) #The R-squared is 0.6291
```

```
##
## Call:
## lm(formula = avg_rating_by_age ~ age_of_movie, data = age_between30_and_60)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.20948 -0.07029 -0.01012  0.06197  0.19994
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.986584   0.106789  27.967  < 2e-16 ***
## age_of_movie 0.015756   0.002333   6.753 2.99e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1051 on 27 degrees of freedom
## Multiple R-squared:  0.6281, Adjusted R-squared:  0.6143
## F-statistic:  45.6 on 1 and 27 DF,  p-value: 2.987e-07
```

```
summary(lm(avg_rating_by_age ~ age_of_movie, data = age_between10_and_50)) #The R-squared is 0.5566
```

```
##
## Call:
## lm(formula = avg_rating_by_age ~ age_of_movie, data = age_between10_and_50)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.22756 -0.07360  0.01328  0.04936  0.21241
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.223619   0.051405  62.710  < 2e-16 ***
## age_of_movie 0.010664   0.001586   6.722 7.63e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1077 on 36 degrees of freedom
## Multiple R-squared:  0.5565, Adjusted R-squared:  0.5442
## F-statistic: 45.18 on 1 and 36 DF,  p-value: 7.633e-08
```

```
summary(lm(avg_rating_by_age ~ age_of_movie, data = age_between20_and_40)) #The R-squared is 0.5188
```

```
##
## Call:
## lm(formula = avg_rating_by_age ~ age_of_movie, data = age_between20_and_40)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.085809 -0.042785  0.006559  0.029602  0.099961
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.174036   0.071118  44.630  < 2e-16 ***
## age_of_movie 0.009943   0.002332   4.264 0.000525 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05568 on 17 degrees of freedom
```

```
## Multiple R-squared:  0.5167, Adjusted R-squared:  0.4883
## F-statistic: 18.18 on 1 and 17 DF,  p-value: 0.0005245
```

```
# Make Figure 3 with Plot5, Plot6, Plot7, and Plot8
grid.arrange(Plot5, Plot6, Plot7, Plot8, nrow=2, ncol=2)
```

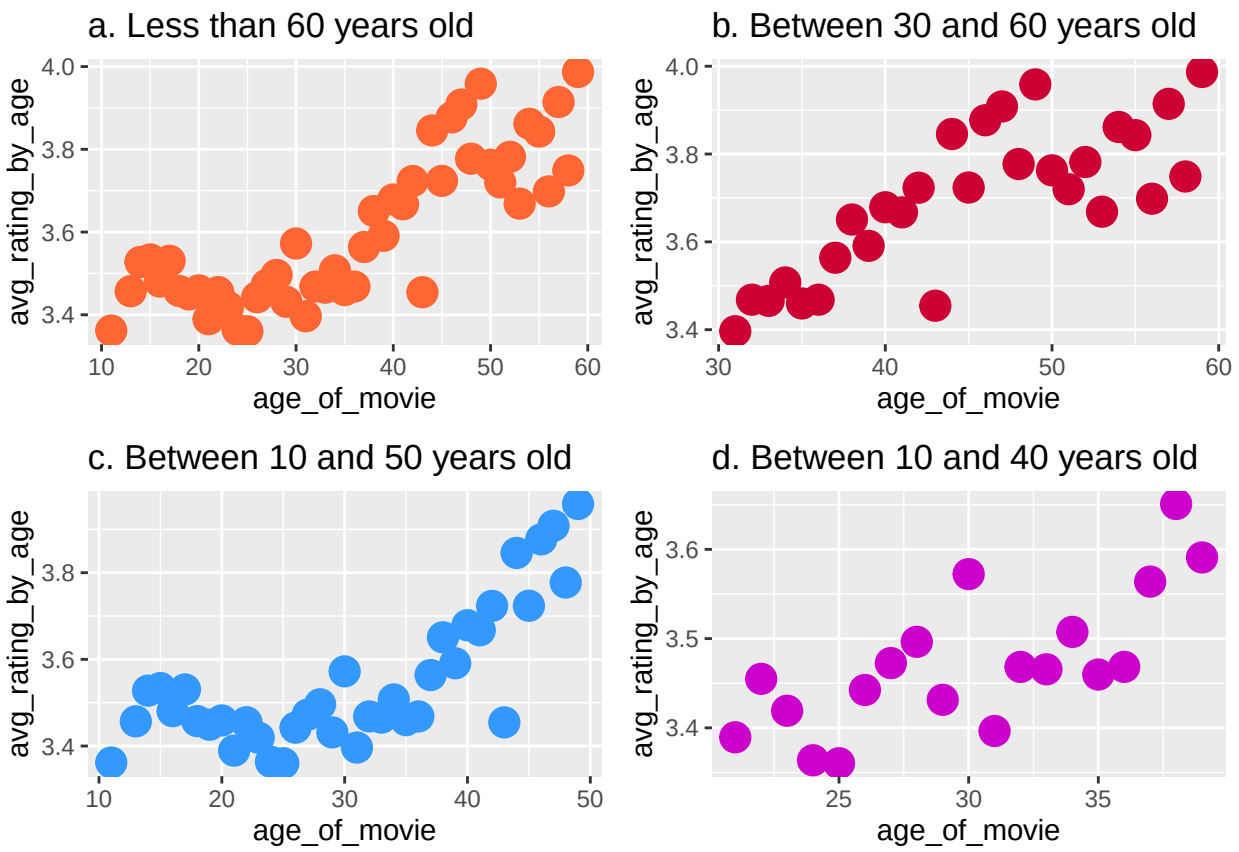


Figure 3. Relationship average rating by age and age of movie.

Genres's effect

```
# Split Genres data into single genres
dat <- edx_with_title_dates %>% separate_rows(genres, sep ="\\|")
head(dat)
```

```
## # A tibble: 6 x 9
##   userId movieId rating title        genres year_rated premier_date age_of_movie
##    <int>   <dbl>  <dbl> <chr>        <chr>       <dbl>        <dbl>        <dbl>
## 1      1     231      5 Dumb & Dumb~ Comedy       1996         1994           27
## 2      1     292      5 Outbreak (1~ Action       1996         1995           26
## 3      1     292      5 Outbreak (1~ Drama        1996         1995           26
## 4      1     292      5 Outbreak (1~ Sci-Fi       1996         1995           26
## 5      1     292      5 Outbreak (1~ Thril~       1996         1995           26
## 6      1     316      5 Stargate (1~ Action       1996         1994           27
## # ... with 1 more variable: rating_date_range <dbl>
```

```r
# Distribution of Ratings according to genres
temp <- dat %>%
  group_by(genres) %>%
  summarize(n=n()) %>%
  ungroup() %>%
  mutate(sumN = sum(n), percentage = n/sumN) %>%
  arrange(-percentage)

# Make bar graph of genres
Plot9 <- temp %>%
  ggplot(aes(reorder(genres, percentage), percentage, fill= percentage)) +
  geom_bar(stat = "identity") + coord_flip() +
  scale_fill_distiller(palette = "Spectral") + labs(y = "Percentage", x = "Genre") +
  ggtitle("a. Distribution of Genres by Percent")

# Make genres's Mean rating
temp <- dat %>%
  group_by(genres) %>%
  summarize(mean_rating_by_genre=mean(rating)) %>%
  arrange(-mean_rating_by_genre)

Plot10 <- temp %>%
  ggplot(aes(reorder(genres, mean_rating_by_genre), mean_rating_by_genre, fill= mean_rating_by_genre)) +
  geom_bar(stat = "identity") + coord_flip() +
  scale_fill_distiller(palette = "Spectral") + labs(y = "Mean Rating", x = "Genre") +
  ggtitle("b. Average Rating of Genres")

# Make Figure 4 with Plot9 and Plot10
grid.arrange(Plot9, Plot10, nrow=2, ncol=1)
```

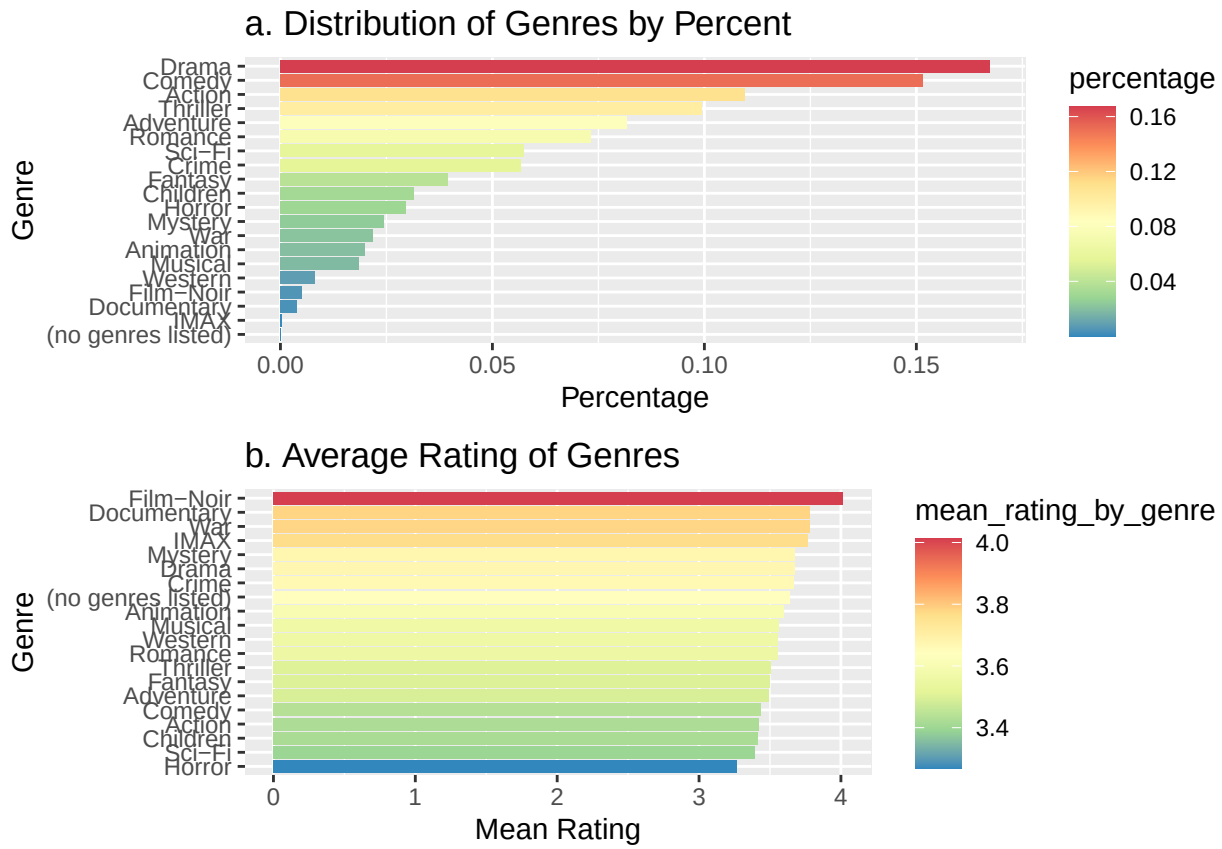


Figure 4. Distribution of genres by percentage (a) and average rating of genres.

Prepare correlation analysis

```
# Make number of movie ratings according to movie
n_movies_ratings <- edx_with_title_dates %>% group_by(movieId) %>% summarize(n = n())

# Make Average Movie Rating for each movie
avg_movie_rat <- edx_with_title_dates %>% group_by(movieId) %>% summarize(avg_m_r = mean(rating))

# Make correlation data
cor_dat <- edx_with_title_dates %>% select(rating, movieId, userId, year_rated, age_of_movie, rating_da
  left_join(n_movies_ratings, by = "movieId") %>%
  left_join(avg_movie_rat, by = 'movieId')
head(cor_dat)
```

```
##   rating movieId userId year_rated age_of_movie rating_date_range premier_date
## 1      5     231      1       1996           27                 2         1994
## 2      5     292      1       1996           26                 1         1995
## 3      5     316      1       1996           27                 2         1994
## 4      5     329      1       1996           27                 2         1994
## 5      5     355      1       1996           27                 2         1994
## 6      5     356      1       1996           27                 2         1994
##       n  avg_m_r
## 1 16056 2.942202
## 2 14350 3.419547
```

```
## 3 17013 3.348910
## 4 14522 3.337901
## 5  4834 2.482830
## 6 31076 4.012469
```

```
# Test pearson correlation analysis
temp <- cor_dat %>% select(one_of("rating", "movieId", "userId", "year_rated", "age_of_movie",
                                  "rating_date_range", "premier_date", "n", "avg_m_r")) %>% as.matrix()
M <- cor(temp)
testRes = cor.mtest(temp, conf.level = 0.95)
corrplot(M, p.mat = testRes$p, method = 'square', type = 'lower', insig='blank',tl.col = 'black',
         addCoef.col ='black', number.cex = 0.8, order = 'hclust', diag=FALSE, col = brewer.pal(n = 10,
```

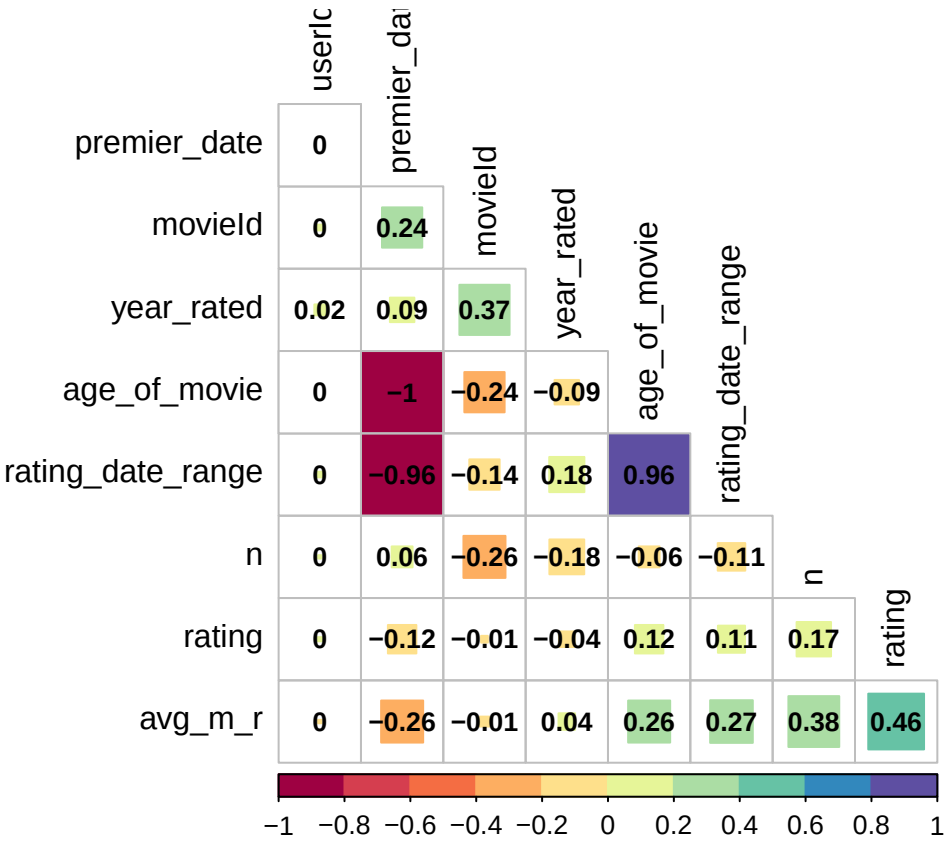
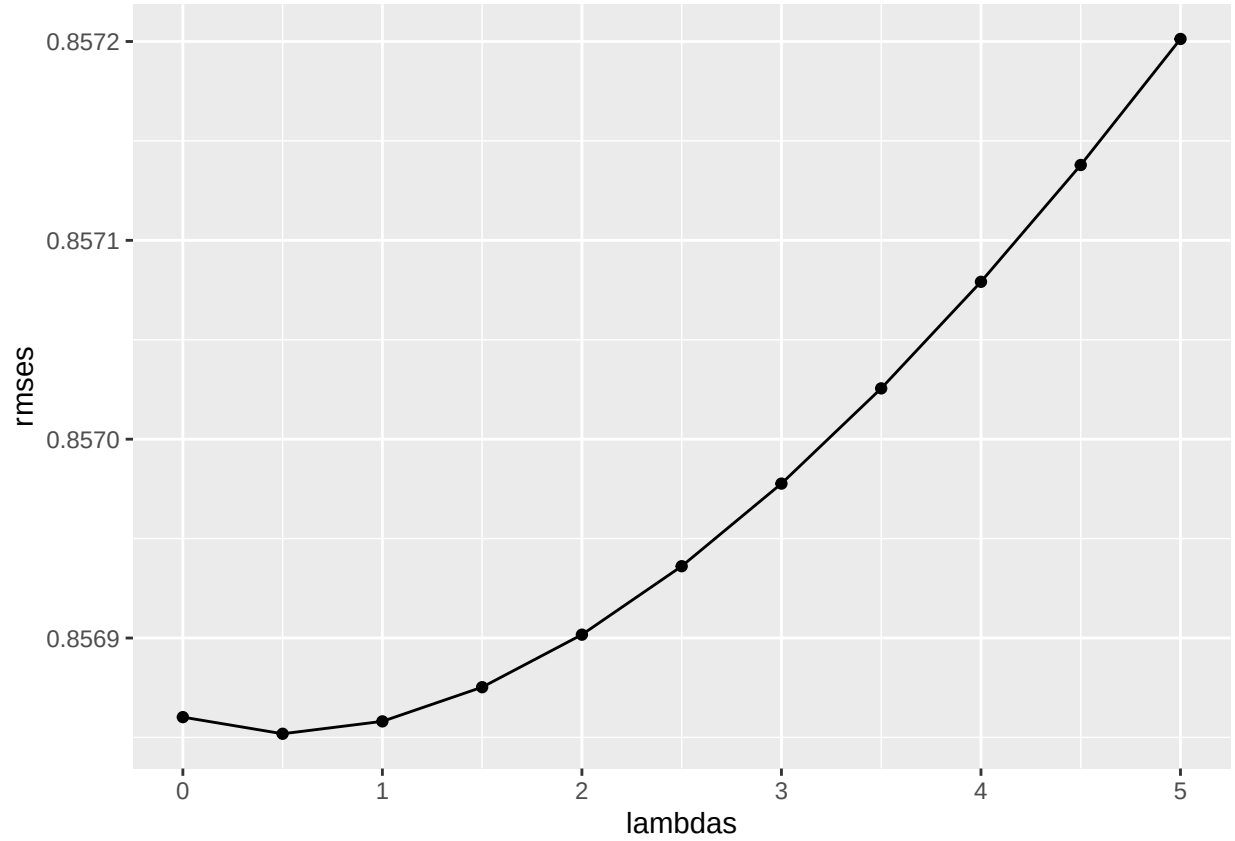


Figure 5. Correlation between ranting, movieID, year rated, age of movie, rating date range, premier data, number of movies ratigs, average movie rate.

Calculate the RMSE

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

# lambda is a tuning parameter
lambdas <- seq(0,5,.5)
```

```r
# For each lambda,find b_i & b_u, followed by rating prediction & testing
rmses <- sapply(lambdas, function(l){
  mu <- mean(edx_with_title_dates$rating)

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

  b_u <- edx_with_title_dates %>%
    left_join(b_i, by='movieId') %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n() +l))

  predicted_ratings <- edx_with_title_dates %>%
    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_with_title_dates$rating))
})

# Plot rmses vs lambdas to select the optimal lambda
qplot(lambdas, rmses, geom=c("point","line"))
```



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

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

```r
# Use the model on the Validation data
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] 0.823484
```

Results and Discussion

In the movielens data, movie ID is 10677 movie, genres is 797, and user ID is 69878 respectively. Distribution of movie rating and distribution of user rating in Movielens data are shown in Figure 1. Relationship between average rating by age and age of movie (a) and average user rating and userID (b) don't show any significant trends (Figure 2). In relationship between average rating according to age of movie, the R-squared value is the highest in lest than 60 years old (Figure 3). In genres's effect, Drama is the highest and IMAX is the lowest in distribution of genres by percent (Figure 4). In Average rating of genres, Film-Noir is the highest and Horror is the lowest. In correlation analysis of movielens data, Premier date and age of movie and premier date and rating date range show significantly negative correlation and age of movie and rating date range shows significantly postivie correlation. In the model on the validation data, RMSE is 8.23484. This imples the prediction of model could be trusted.