# MovieLens

Shuji Hachisu

13/02/2022

## Executive Summary

This report explores ratings of movies given my million of users. The ratings will be analysed with respect to movie title, movie release year, genre and individuals providing the rating. A machine learning (ML) movie recommendation system was created with a room mean square error (RMSE) of 0.8643. This high accuracy was achieved by using a regularization techniques and taking the following effects into account: movie, user, genre and release year.

## Method and Analysis

In this project, a movie recommendation system will be created by applying ML principles to publicly available ratings of movies called MovieLens. The ratings are given on a scale of 1 to 5 with 1 being the worst and 5 being the best rating. The full MovieLens dataset is 27 million records and can be found here: https://grouplens.org/datasets/movielens/latest/. This project will use a subset of the data, 10 million records.

### Create training (a.k.a edx) and validation sets

MovieLens data set will be downloaded from the web, and the files will be read and loaded into R as a data frame. The data frame will be given appropriate column names, and some field will be further processed to separate useful data: movie titles and release year. Timestamp will also be processed appropriately to allow convenient handling of when the ratings were given. The data set will be split into training (aka edx) and validation set to build the model and test the model respectively.

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

# Note: this process could take a couple of minutes

# install packages if needed

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")

# load libraries
```

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

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

# download movielens data set and save to temprorary file

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

# read ratings.dat file and store in ratings

ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                 col.names = c("userId", "movieId", "rating", "timestamp"))

# read movies.dat file and store in movies

movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)

# set name of columns/fields in movies

colnames(movies) <- c("movieId", "title", "genres")

# if using R 3.6 or earlier:
# type cast column
#movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
#                                           title = as.character(title),
#                                           genres = as.character(genres))

# if using R 4.0 or later:
# type cast column
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))

# join ratings and movies using movieId as key

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

# Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)

# slice movielens to create training set (edx) and temp set

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)

# remove unnecessary variable and data sets
rm(dl, ratings, movies, test_index, temp, movielens, removed)

# establish release year. process time stamp. create fields for year and month ratings were made
edx <-
  edx %>%
  mutate(rating_time =
           as.Date(as.POSIXct(timestamp, origin = "1970-01-01")),
         rating_year = year(rating_time),
         rating_month = month(rating_time),
         release_year = as.integer(
           substr(title, str_length(title) - 4, str_length(title) - 1)
         )) %>%
  select(-timestamp, -rating_time)

validation <-
  validation %>%
  mutate(rating_time =
           as.Date(as.POSIXct(timestamp, origin = "1970-01-01")),
         rating_year = year(rating_time),
         rating_month = month(rating_time),
         release_year = as.integer(
           substr(title, str_length(title) - 4, str_length(title) - 1)
         )) %>%
  select(-timestamp, -rating_time)
```

## Explore and analyse the data set and its features

The first 6 rows of the data tables and their corresponding summary statistics are shown to gain insights into both the edx and validaiton sets.

```
#show first 6 rows of the training set (edx) to get insight into the table and data structure
head(edx)
```

```
##    userId movieId rating                         title
## 1:      1     122      5              Boomerang (1992)
## 2:      1     185      5               Net, The (1995)
## 3:      1     292      5               Outbreak (1995)
## 4:      1     316      5               Stargate (1994)
## 5:      1     329      5 Star Trek: Generations (1994)
## 6:      1     355      5       Flintstones, The (1994)
##                           genres rating_year rating_month release_year
## 1:                Comedy|Romance        1996            8         1992
## 2:         Action|Crime|Thriller        1996            8         1995
## 3:  Action|Drama|Sci-Fi|Thriller        1996            8         1995
```

```
## 4:       Action|Adventure|Sci-Fi        1996            8         1994
## 5: Action|Adventure|Drama|Sci-Fi        1996            8         1994
## 6:       Children|Comedy|Fantasy        1996            8         1994
```

```
# show summary statistics of edx data
summary(edx)
```

```
##      userId         movieId          rating         title
##  Min.   :    1   Min.   :    1   Min.   :0.500   Length:9000055
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   Class :character
##  Median :35738   Median : 1834   Median :4.000   Mode  :character
##  Mean   :35870   Mean   : 4122   Mean   :3.512
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000
##  Max.   :71567   Max.   :65133   Max.   :5.000
##     genres           rating_year    rating_month     release_year
##  Length:9000055     Min.   :1995   Min.   : 1.000   Min.   :1915
##  Class :character   1st Qu.:2000   1st Qu.: 4.000   1st Qu.:1987
##  Mode  :character   Median :2002   Median : 7.000   Median :1994
##                     Mean   :2002   Mean   : 6.786   Mean   :1990
##                     3rd Qu.:2005   3rd Qu.:10.000   3rd Qu.:1998
##                     Max.   :2009   Max.   :12.000   Max.   :2008
```

```
# edx data set has 9,000,055 records and 6 fields

#show first 6 rows of the validation set to get insight into the table and data structure
head(validation)
```

```
##    userId movieId rating
## 1:      1     231      5
## 2:      1     480      5
## 3:      1     586      5
## 4:      2     151      3
## 5:      2     858      2
## 6:      2    1544      3
##                                                 title
## 1:                                 Dumb & Dumber (1994)
## 2:                                 Jurassic Park (1993)
## 3:                                    Home Alone (1990)
## 4:                                       Rob Roy (1995)
## 5:                                 Godfather, The (1972)
## 6: Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
##                                    genres rating_year rating_month
## 1:                                 Comedy        1996            8
## 2:        Action|Adventure|Sci-Fi|Thriller        1996            8
## 3:                         Children|Comedy        1996            8
## 4:                Action|Drama|Romance|War        1997            7
## 5:                            Crime|Drama        1997            7
## 6: Action|Adventure|Horror|Sci-Fi|Thriller        1997            7
##    release_year
## 1:         1994
## 2:         1993
## 3:         1990
## 4:         1995
```

```
## 5:         1972
## 6:         1997
```

```
# show summary statistics of validation data
summary(validation)
```

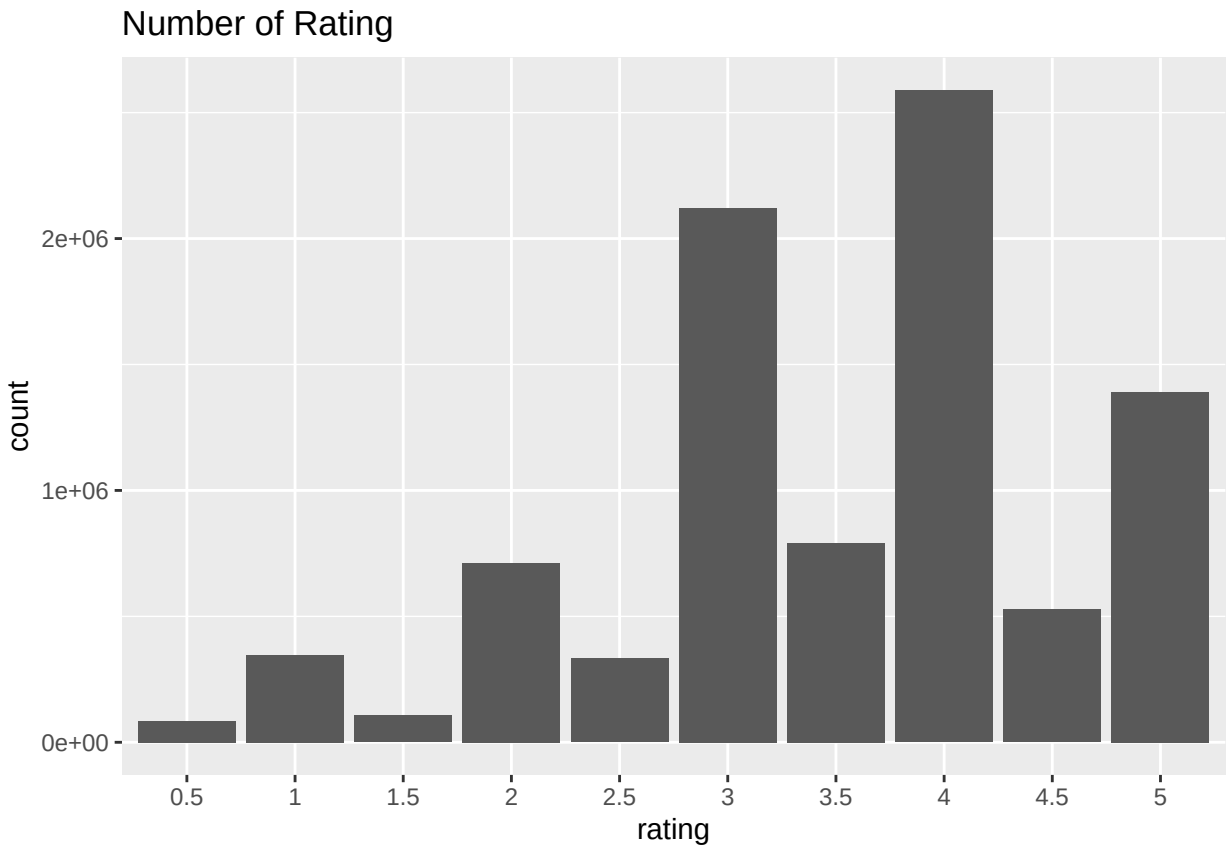
```
##      userId         movieId          rating         title
##  Min.   :    1   Min.   :    1   Min.   :0.500   Length:999999
##  1st Qu.:18096   1st Qu.:  648   1st Qu.:3.000   Class :character
##  Median :35768   Median : 1827   Median :4.000   Mode  :character
##  Mean   :35870   Mean   : 4108   Mean   :3.512
##  3rd Qu.:53621   3rd Qu.: 3624   3rd Qu.:4.000
##  Max.   :71567   Max.   :65133   Max.   :5.000
##     genres           rating_year    rating_month     release_year
##  Length:999999      Min.   :1995   Min.   : 1.000   Min.   :1915
##  Class :character   1st Qu.:1999   1st Qu.: 4.000   1st Qu.:1987
##  Mode  :character   Median :2002   Median : 7.000   Median :1994
##                     Mean   :2002   Mean   : 6.784   Mean   :1990
##                     3rd Qu.:2005   3rd Qu.:10.000   3rd Qu.:1998
##                     Max.   :2009   Max.   :12.000   Max.   :2008
```

```
# Validation data set has 999,999 records and 6 fields.
# This will be used at the final stage to evaluated the model created using the edx data set
```

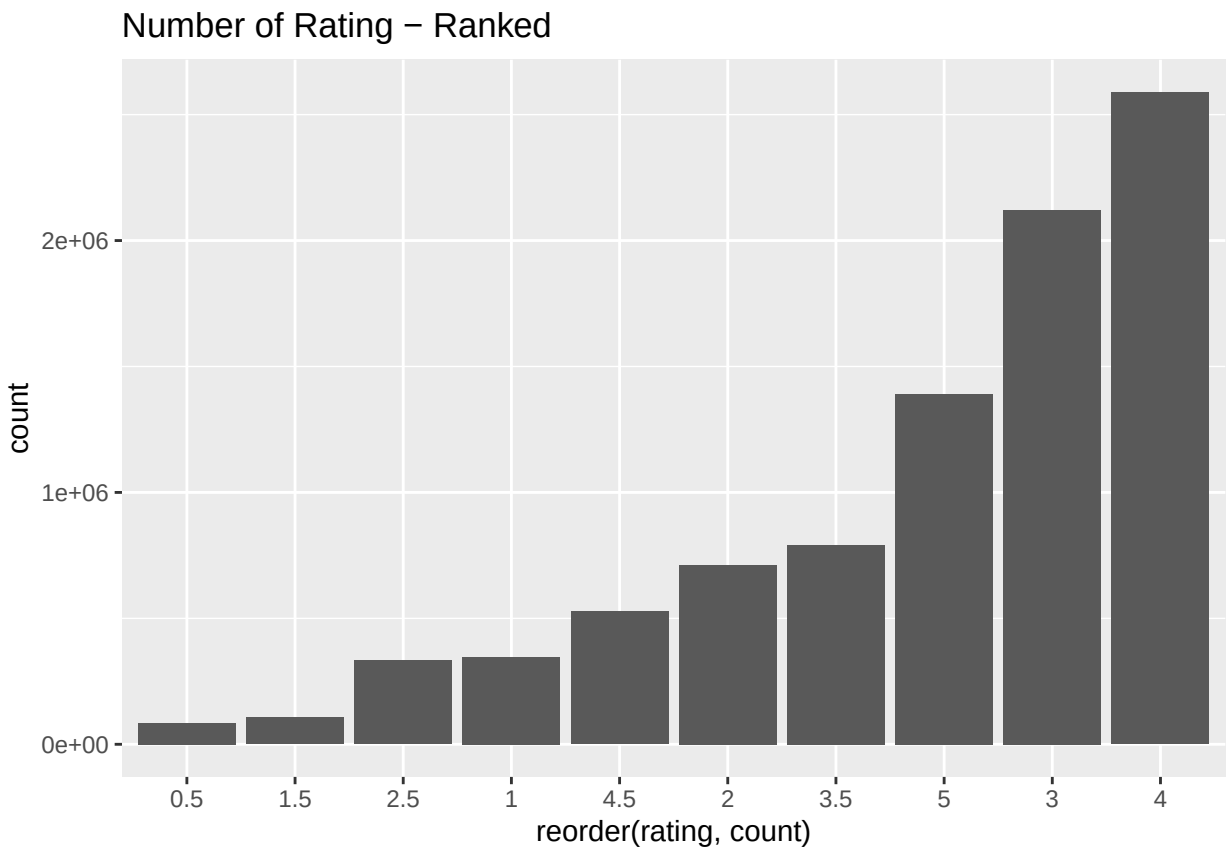
### Analyze frequency of ratings

The plots derived by following R codes allows exploration of how popular certain rating values are. The first plot shows that in general whole number rankings are more popular than decimal ratings such as 0.5, 1.5, 2.5, 3.5 and 4.5. The second plot ranks the ratings in order of popularity, which shows that the 5 most common ratings are from the highest to lowest: 4, 3, 5, 3.5 and 2.

```
######################   Analyze frequency of ratings   ########################


# create table that describes total number ratings received per rating

number_of_rating <-
  edx %>%
  group_by(rating) %>%
  summarize(count = n()) %>%
  mutate(rating = factor(rating))

# plot rating vs count for that rating

number_of_rating %>%
  ggplot(aes(rating, count)) +
  geom_col() +
  labs(title = "Number of Rating")
```




```
# Plot ranking of rating

number_of_rating %>%
  ggplot(aes(reorder(rating,count), count)) +
  geom_col() +
  labs(title = "Number of Rating - Ranked")
```



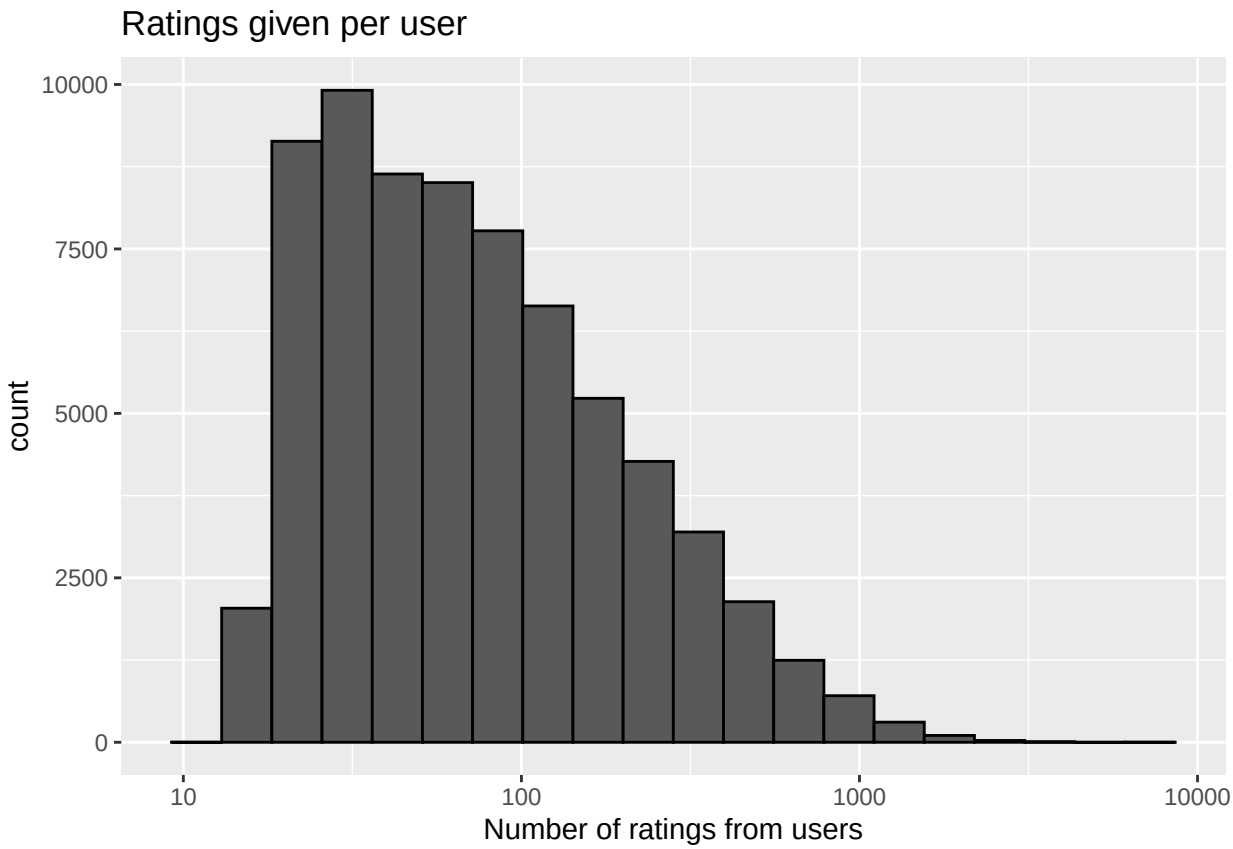

**Analyze ratings by userId**

The first plot and the immediately following summary statistics table shows how often each users provide ratings to movies. The average user gives c. 129 ratings, most prolific users have given 6616 ratings, the most inactive users give as little as 10 ratings, and 3 quarter of the people give less than 141 ratings.

There are also 2 tables that show the top 10 and the least active 10 users and the average rating they gave out along with the standard deviation of the ratings. The to 10 users have tight means ranging from 2.4-3.8 with low standard deviation ranging from 0.6-1.0. The bottom 10 active users have a wide variety of means ranging from 1.8-4.30 with also a wide variety of standard deviation ranging from 0.5-1.4.

The final plot of this section shows that users that give low number of ratings on average give higher ratings at around a mean of 3.5, and the the prolific raters give out a lower mean ratings coverging to 3.2.

```
######################   Analyze ratings by userId   ############################

# Histogram of number of rating given per user

edx %>%
  count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 20, color = "black") +
  scale_x_log10() +
  labs(title = "Ratings given per user",
       x = "Number of ratings from users")
```




```
# create table that describes rating activity of each user
user_summary <-
  edx %>%
  group_by(userId) %>%
  summarize(user_number_rated = n(),
            user_mu = mean(rating),
            user_sd = sd(rating))

# show summary statistics of table created above describing rating activity of users
summary(user_summary)
```

```
##      userId      user_number_rated    user_mu         user_sd
##  Min.   :    1   Min.   :  10.0    Min.   :0.500   Min.   :0.0000
##  1st Qu.:17943   1st Qu.:  32.0    1st Qu.:3.357   1st Qu.:0.8012
##  Median :35799   Median :  62.0    Median :3.635   Median :0.9388
##  Mean   :35782   Mean   : 128.8    Mean   :3.614   Mean   :0.9598
##  3rd Qu.:53620   3rd Qu.: 141.0    3rd Qu.:3.903   3rd Qu.:1.0980
##  Max.   :71567   Max.   :6616.0    Max.   :5.000   Max.   :2.3152
```

```
# Top 10 users who has given the most ratings. The max is 6616 rankings by a user
user_summary %>%
  arrange(desc(user_number_rated)) %>%
  head(10)
```

```
## # A tibble: 10 x 4
```

```
##    userId user_number_rated user_mu user_sd
##     <int>             <int>   <dbl>   <dbl>
##  1  59269              6616    3.26   0.639
##  2  67385              6360    3.20   0.957
##  3  14463              4648    2.40   0.688
##  4  68259              4036    3.58   1.05
##  5  27468              4023    3.83   0.734
##  6  19635              3771    3.50   0.778
##  7   3817              3733    3.11   0.579
##  8  63134              3371    3.27   0.957
##  9  58357              3361    3.00   0.798
## 10  27584              3142    3.00   0.719
```

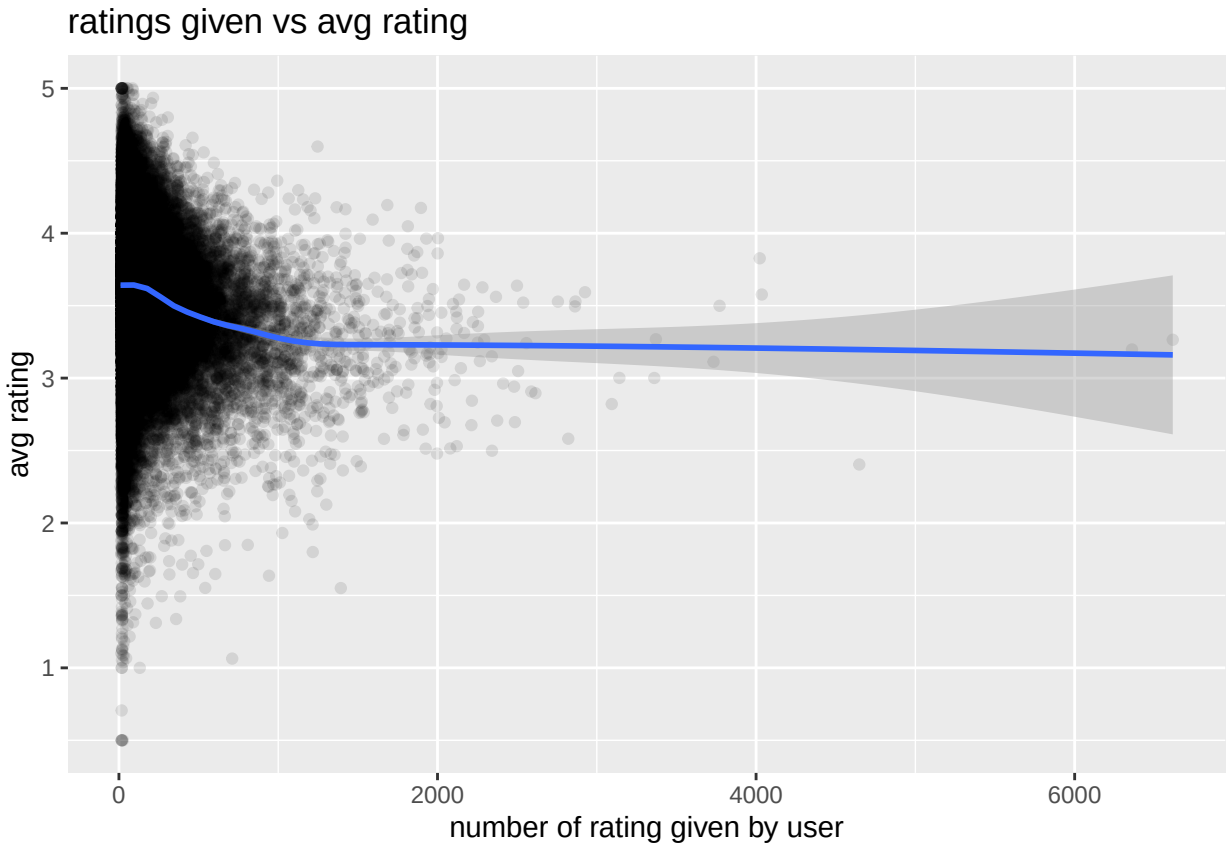
```
# Bottom 10 users who has given the least ratings. The min is 10 rankings by a useer
user_summary %>%
  arrange(user_number_rated) %>%
  head(10)
```

```
## # A tibble: 10 x 4
##    userId user_number_rated user_mu user_sd
##     <int>             <int>   <dbl>   <dbl>
##  1  62516                10    2.25   1.14
##  2  22170                12    4      0.739
##  3  15719                13    3.77   1.24
##  4  50608                13    3.92   1.44
##  5    901                14    4.71   0.469
##  6   1833                14    3      1.24
##  7   2476                14    2.93   1.33
##  8   5214                14    1.79   1.22
##  9   9689                14    3.57   1.22
## 10  10364                14    4.32   1.27
```

```
# average number of ratings provider per user is 129
mean(user_summary$user_number_rated)
```

```
## [1] 128.7967
```

```
# plot number of rating given by user vs average rating given by that user
user_summary %>%
  ggplot(aes(user_number_rated, user_mu)) +
  geom_point(alpha = 0.1) +
  geom_smooth() +
  labs(x = "number of rating given by user",
       y = "avg rating",
       title = "ratings given vs avg rating")
```



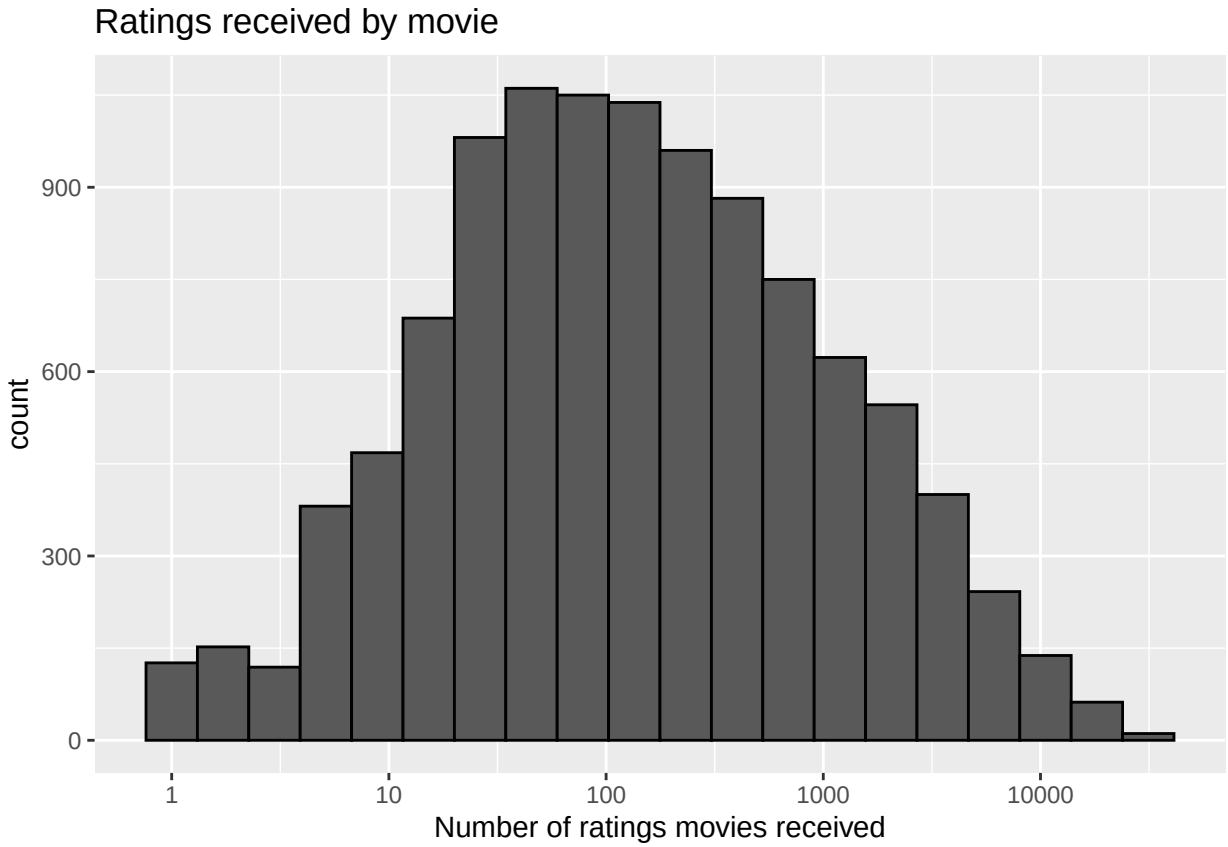

**Analyze ratings by movieId**

The first plot and the immediately following summary statistics table shows how often each movies are rated. On average movies get c. 843 ratings, most rated movie has 31362 ratings, the least rated movie has 1 rating, and 3 quarter of movies get less than 565 ratings.

There are also 2 tables that show the top 10 and the least 10 rated movies, and the average rating they revived along with the standard deviation of the ratings. The to 10 movies have tight means ranging from 3.7-4.2 with low standard deviation ranging from 0.7-1.0. The least rated movies have a wide variety of means ranging from 1-5 with no standard deviation as all of these films only received one rating.

The final plot of this section shows that movies that get low number of ratings on average get lower ratings at around a mean of 3, and most rated films get higher ratings at around a mean of 4.

```
######################   Analyze ratings by movieId   ############################


# histogram of number of rating per movie

edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 20, color = "black") +
  scale_x_log10() +
  labs(title = "Ratings received by movie",
       x = "Number of ratings movies received")
```




```
# create table that describes how often each moving is rated and its rating
movie_summary <-
  edx %>%
  group_by(movieId) %>%
  summarize(movie_number_rated = n(),
            movie_mu = mean(rating),
            movie_sd = sd(rating))

# show summary statistics of table created above describing rating activity of movies
summary(movie_summary)
```

```
##     movieId      movie_number_rated    movie_mu        movie_sd
##  Min.   :    1   Min.   :    1.0    Min.   :0.500   Min.   :0.0000
##  1st Qu.: 2754   1st Qu.:   30.0    1st Qu.:2.844   1st Qu.:0.8476
##  Median : 5434   Median :  122.0    Median :3.268   Median :0.9510
##  Mean   :13105   Mean   :  842.9    Mean   :3.192   Mean   :0.9540
##  3rd Qu.: 8710   3rd Qu.:  565.0    3rd Qu.:3.609   3rd Qu.:1.0619
##  Max.   :65133   Max.   :31362.0    Max.   :5.000   Max.   :2.4749
##                                                     NA's   :126
```

```
# Top 10 movies that have received the most ratings
movie_summary %>%
  arrange(desc(movie_number_rated)) %>%
  head(10)
```

```
## # A tibble: 10 x 4
```

```
##    movieId movie_number_rated movie_mu movie_sd
##      <dbl>              <int>    <dbl>    <dbl>
##  1     296              31362     4.15    1.00
##  2     356              31079     4.01    0.972
##  3     593              30382     4.20    0.839
##  4     480              29360     3.66    0.938
##  5     318              28015     4.46    0.717
##  6     110              26212     4.08    0.953
##  7     457              25998     4.01    0.778
##  8     589              25984     3.93    0.906
##  9     260              25672     4.22    0.914
## 10     150              24284     3.89    0.852
```

```
# Bottom 10 movies that have received the least ratings
movie_summary %>%
  arrange(movie_number_rated) %>%
  head(10)
```

```
## # A tibble: 10 x 4
##    movieId movie_number_rated movie_mu movie_sd
##      <dbl>              <int>    <dbl>    <dbl>
##  1    3191                  1      3.5       NA
##  2    3226                  1      5         NA
##  3    3234                  1      3         NA
##  4    3356                  1      3         NA
##  5    3383                  1      3         NA
##  6    3561                  1      1         NA
##  7    3583                  1      3         NA
##  8    4071                  1      1         NA
##  9    4075                  1      1         NA
## 10    4820                  1      2         NA
```

```
# average number of ratings provide per movie is 843
mean(movie_summary$movie_number_rated)
```

```
## [1] 842.9386
```

```
# plot number of rating recevied by movie and its average rating
movie_summary %>%
  ggplot(aes(movie_number_rated, movie_mu)) +
  geom_point(alpha = 0.1) +
  geom_smooth() +
  labs(x = "number of rating recevied by movies",
       y = "avg rating",
       title = "ratings recieved vs avg rating")
```


**Analyze ratings by release year**

The first plot and summary statistics table shows how rating of the movies have varied with movie release year. On average, each movie release year has 95746 ratings in total. The most rated release year has 786762 ratings during the early 90s and least rated moive release year has only 32 ratings.

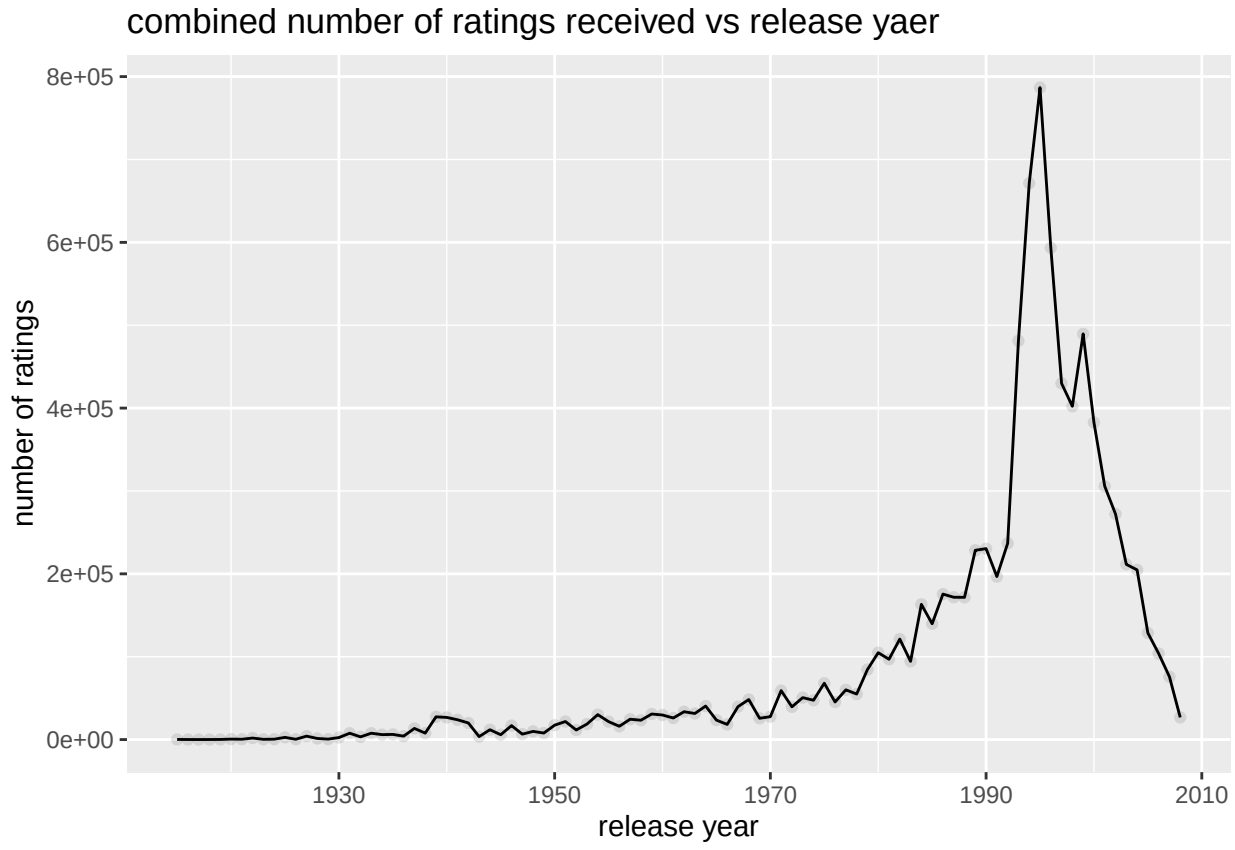
The second plot shows that movies released between 1940-1960 have high average rating around 3.9, where as movies released post 1990 average rating less than 3.5 and approaching 3.4 by 2010 release year.

```
################   Analyze ratings by release year   ################


# create table that describes how movies perform for rating based on release year

release_year_summary <-
  edx %>%
  group_by(release_year) %>%
  summarize(release_year_number_of_rating  = n(),
            release_year_mu = mean(rating),
            release_year_sd = sd(rating))

# show summary statistics of table created above describing rating variation by release year
summary(release_year_summary)
```

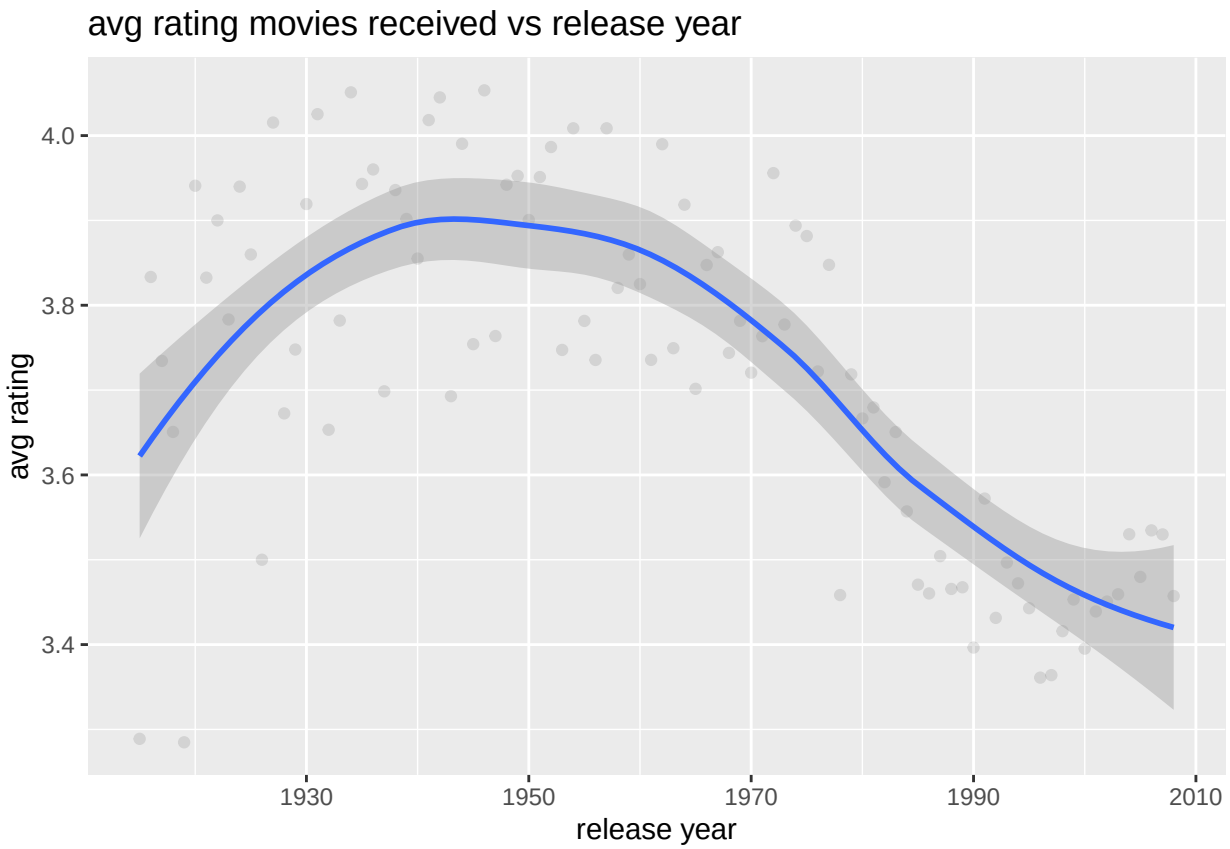
```
##   release_year  release_year_number_of_rating release_year_mu release_year_sd
```

```
##  Min.   :1915   Min.   :    32              Min.   :3.285   Min.   :0.8431
##  1st Qu.:1938   1st Qu.:  7670              1st Qu.:3.511   1st Qu.:0.9404
##  Median :1962   Median : 27082              Median :3.748   Median :0.9825
##  Mean   :1962   Mean   : 95745              Mean   :3.721   Mean   :0.9898
##  3rd Qu.:1985   3rd Qu.:104599              3rd Qu.:3.900   3rd Qu.:1.0471
##  Max.   :2008   Max.   :786762              Max.   :4.053   Max.   :1.1912
```

```
# plot release year of movies vs number of ratings received by movies during that year
release_year_summary %>%
  ggplot(aes(release_year, release_year_number_of_rating)) +
  geom_point(alpha = 0.1) +
  geom_line() +
  labs(x = "release year",
       y = "number of ratings",
       title = "combined number of ratings received vs release yaer")
```




```
# plot release year of movies vs average ratings received by movies during that year
release_year_summary %>%
  ggplot(aes(release_year, release_year_mu)) +
  geom_point(alpha = 0.1) +
  geom_smooth() +
  labs(x = "release year",
       y = "avg rating",
       title = "avg rating movies received vs release year")
```




**Analyze rating by genre**

The first table shows summary statistics that shows min, max, median and quadrilles of total number of ratings received by each composite movive genres.

The second table shows the top 10 most rated genres and corresponding average rating and associated standard deviation. The top genres making up the top 6 composites genres are: Drama, comedy and romance. The addition of action, adventures, Sci-Fi, thriller and crime captures all 10 most rated composite genres. The mean ratings range from 3.2-3.9 with a standard deviation ranging 0.9-1.1.

The third table shows the least 10 rated genres and corresponding average rating and associated standard deviation. The mean ratings range from 1.5-4.0 with a standard deviation ranging 0-1.4 as these only have 2 to 3 ratings making the ratings hihgly variable.

The fourth table shows the top 10 rated genres. The highest average rating received by a composite genre being Animation|IMAX|Sci-Fi, but this composite genre has only received 7 ratings in total. The second highest rated composite genre is Drama|Film-Noir|Romance, which has close to 3000 rankings, and is more significant than Animation|IMAX|Sci-Fi, which is an outlier generated by low number of ratings.

The fifth table shows the worst 10 rate genres and Documentary|Horror has recived an average rating of 1.4 with a significant number of ratings: 619.

```
#########################   Analyze rating by genre   ############################

# create table that describes how movies perform for rating based on genre

genre_summary <-
```

```
  edx %>%
  group_by(genres) %>%
  summarize(genres_number_of_rating  = n(),
            genre_mu = mean(rating),
            genre_sd = sd(rating))

# show summary statistics of table created above describing rating variation by genre
summary(genre_summary)
```

```
##     genres          genres_number_of_rating    genre_mu        genre_sd
##  Length:797         Min.   :     2          Min.   :1.449   Min.   :0.0000
##  Class :character   1st Qu.:   185          1st Qu.:3.116   1st Qu.:0.8945
##  Mode  :character   Median :  1459          Median :3.437   Median :0.9806
##                     Mean   : 11292          Mean   :3.360   Mean   :0.9735
##                     3rd Qu.:  7167          3rd Qu.:3.694   3rd Qu.:1.0686
##                     Max.   :733296          Max.   :4.714   Max.   :1.7536
```

```
# display a table showing the most rated genres
genre_summary %>%
  arrange(desc(genres_number_of_rating)) %>%
  head(10)
```

```
## # A tibble: 10 x 4
##    genres                    genres_number_of_rating genre_mu genre_sd
##    <chr>                                      <int>    <dbl>    <dbl>
##  1 Drama                                     733296     3.71    0.980
##  2 Comedy                                    700889     3.24    1.11
##  3 Comedy|Romance                            365468     3.41    1.03
##  4 Comedy|Drama                              323637     3.60    0.994
##  5 Comedy|Drama|Romance                      261425     3.65    0.981
##  6 Drama|Romance                             259355     3.61    1.03
##  7 Action|Adventure|Sci-Fi                   219938     3.51    1.09
##  8 Action|Adventure|Thriller                 149091     3.43    0.951
##  9 Drama|Thriller                            145373     3.45    0.960
## 10 Crime|Drama                               137387     3.95    0.903
```

```
# display a table showing the least rated genres
genre_summary %>%
  arrange(genres_number_of_rating) %>%
  head(10)
```

```
## # A tibble: 10 x 4
##    genres                                genres_number_of_rat~ genre_mu genre_sd
##    <chr>                                                 <int>    <dbl>    <dbl>
##  1 Action|Animation|Comedy|Horror                            2      1.5    0.707
##  2 Action|War|Western                                        2      3.75   0.354
##  3 Adventure|Fantasy|Film-Noir|Mystery|~                     2      4      1.41
##  4 Adventure|Mystery                                         2      3.25   0.354
##  5 Crime|Drama|Horror|Sci-Fi                                 2      3.25   0.354
##  6 Documentary|Romance                                       2      3.75   1.06
##  7 Drama|Horror|Mystery|Sci-Fi|Thriller                      2      3.5    0
##  8 Fantasy|Mystery|Sci-Fi|War                                2      2.5    0
```

```
##  9 Action|Adventure|Animation|Comedy|Sc~                    3     4       0.866
## 10 Horror|War|Western                                       3     3.33    0.764
```

```
# display a table showing the most highly ranked genres
genre_summary %>%
  arrange(desc(genre_mu)) %>%
  head(10)
```

```
## # A tibble: 10 x 4
##    genres                                genres_number_of_rati~ genre_mu genre_sd
##    <chr>                                                  <int>    <dbl>    <dbl>
##  1 Animation|IMAX|Sci-Fi                                      7     4.71    0.567
##  2 Drama|Film-Noir|Romance                                 2989     4.30    0.791
##  3 Action|Crime|Drama|IMAX                                 2353     4.30    0.739
##  4 Animation|Children|Comedy|Crime                          7167     4.28    0.815
##  5 Film-Noir|Mystery                                       5988     4.24    0.788
##  6 Crime|Film-Noir|Mystery                                 4029     4.22    0.762
##  7 Film-Noir|Romance|Thriller                              2453     4.22    0.738
##  8 Crime|Film-Noir|Thriller                                4844     4.21    0.830
##  9 Crime|Mystery|Thriller                                 26892     4.20    0.844
## 10 Action|Adventure|Comedy|Fantasy|Rom~                   14809     4.20    0.862
```

```
# display a table showing the worst ranked genres
genre_summary %>%
  arrange(genre_mu) %>%
  head(10)
```

```
## # A tibble: 10 x 4
##    genres                                 genres_number_of_rat~ genre_mu genre_sd
##    <chr>                                                  <int>    <dbl>    <dbl>
##  1 Documentary|Horror                                       619     1.45    1.20
##  2 Action|Animation|Comedy|Horror                             2     1.5     0.707
##  3 Action|Horror|Mystery|Thriller                           327     1.61    1.10
##  4 Comedy|Film-Noir|Thriller                                 21     1.64    0.868
##  5 Action|Drama|Horror|Sci-Fi                                 4     1.75    1.26
##  6 Adventure|Drama|Horror|Sci-Fi|Thrill~                    217     1.75    1.09
##  7 Action|Adventure|Drama|Fantasy|Sci-Fi                     57     1.90    1.08
##  8 Action|Children|Comedy                                   518     1.91    1.18
##  9 Action|Adventure|Children                                824     1.92    1.25
## 10 Adventure|Animation|Children|Fantasy~                    691     1.92    1.20
```

# Results

Movie recommendation models are now created with increasing complexity and sophistication starting from an ML model that simply predicts the average rating to an enhanced model that takes into account the following effects: movie, user, release year, genres and regularization.

## Create recommendation models and calcualte RMSEs

Define a function that calculates root mean square error (RMSE) trhough the following input: actual and predicted ratings.

```
################################################
# Create recommendation models and calcualte RMSEs
################################################

# create function for calculating root mean square error (RMSE)
RMSE <- function (true_ratings, predicted_ratings) {
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

## Model 1: based on just using the avg rating

The first model assumes that all users simply give out the average rating. The RMSE is very high and greater than 1. This is neither a sophisticated nor good model.

```
################# Model 1: based on just using the avg rating ##################

# create a simple model that predicts average rating all the time
mu_hat <- mean(edx$rating)

# calculate RMSE of model
naive_rmse <- RMSE(validation$rating, mu_hat)

# load results to a table summarizing the various models
rmse_results <-
  data_frame(method = "Just the average", RMSE = naive_rmse)

# print out all the RMSEs for all the models explored above
rmse_results %>% knitr::kable()
```

| method | RMSE |
|---|---|
| Just the average | 1.061202 |

## Model 2: account for movie effect

The second model take into account individual movie effect, and lowers RMSE to 0.9439.

```
################# Model 2: account for movie effect ############################

# store average rating in mu
mu <- mean(edx$rating)

# calculate movie effect and store result in table
movie_avgs <-
  edx %>%
  group_by(movieId) %>%
  summarize(b_movie = mean(rating - mu))

# predict rating of movies taking into account the above calculated movie effect
predicted_ratings_movie <-
  mu +
```

```
  validation %>%
  left_join(movie_avgs, by = 'movieId') %>%
  .$b_movie

# calculate RMSE of model
model_movie_rmse <- RMSE(validation$rating, predicted_ratings_movie)

# load results to a table summarizing the various models
rmse_results <-
  bind_rows(rmse_results,
            data_frame(method = "Movie Effect Model",
                       RMSE = model_movie_rmse))

# print out all the RMSEs for all the models explored above
rmse_results %>% knitr::kable()
```

| method | RMSE |
|---|---|
| Just the average | 1.0612018 |
| Movie Effect Model | 0.9439087 |

### Model 3: account for user effect

The third model take into account individual user effect in addition movie effect, and further lowers RMSE to 0.8653.

```
######################## Model 3: account for user effect ########################

# calculate user effect and store result in table
user_avgs <-
  edx %>%
  left_join(movie_avgs, by = 'movieId') %>%
  group_by(userId) %>%
  summarize(b_user = mean(rating - mu - b_movie))

# predict rating of movies taking into account the above calculated user effect
predicted_ratings_movie_user <-
  validation %>%
  left_join(movie_avgs, by = 'movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  mutate(pred = mu + b_movie + b_user) %>%
  .$pred

# calculate RMSE of model
model_movie_user_rmse <- RMSE(validation$rating, predicted_ratings_movie_user)

# load results to a table summarizing the various models
rmse_results <-
  bind_rows(rmse_results,
            data_frame(method = "Movie+User Effect Model",
                       RMSE = model_movie_user_rmse))
```

```
# print out all the RMSEs for all the models explored above
rmse_results %>% knitr::kable()
```

| method | RMSE |
|---|---|
| Just the average | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| Movie+User Effect Model | 0.8653488 |

### Model 4: account for release year effect

The fourth model take into account release year of the movie in addition to movie and user effect, and further lowers RMSE to 0.8650.

```
######################## Model 4: account for release year effect ###############

# calculate release year effect and store result in table
year_avgs <-
  edx %>%
  left_join(movie_avgs, by = 'movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  group_by(release_year) %>%
  summarize(b_year = mean(rating - mu - b_movie - b_user))

# predict rating of movies taking into account the above calculated release year effect
predicted_ratings_movie_user_year <-
  validation %>%
  left_join(movie_avgs, by = 'movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  left_join(year_avgs, by = 'release_year') %>%
  mutate(pred = mu + b_movie + b_user + b_year) %>%
  .$pred

# calculate RMSE of model
model_movie_user_year_rmse <- RMSE(validation$rating, predicted_ratings_movie_user_year)

# load results to a table summarizing the various models
rmse_results <-
  bind_rows(rmse_results,
            data_frame(method = "Movie+User+Year Effect Model",
                       RMSE = model_movie_user_year_rmse))

# print out all the RMSEs for all the models explored above
rmse_results %>% knitr::kable()
```

| method | RMSE |
|---|---|
| Just the average | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| Movie+User Effect Model | 0.8653488 |
| Movie+User+Year Effect Model | 0.8650043 |

### Model 5: account for genre effect

The fifth model take into account genere effect in addition to movie, user and release year effect, and further lowers RMSE to 0.8647.

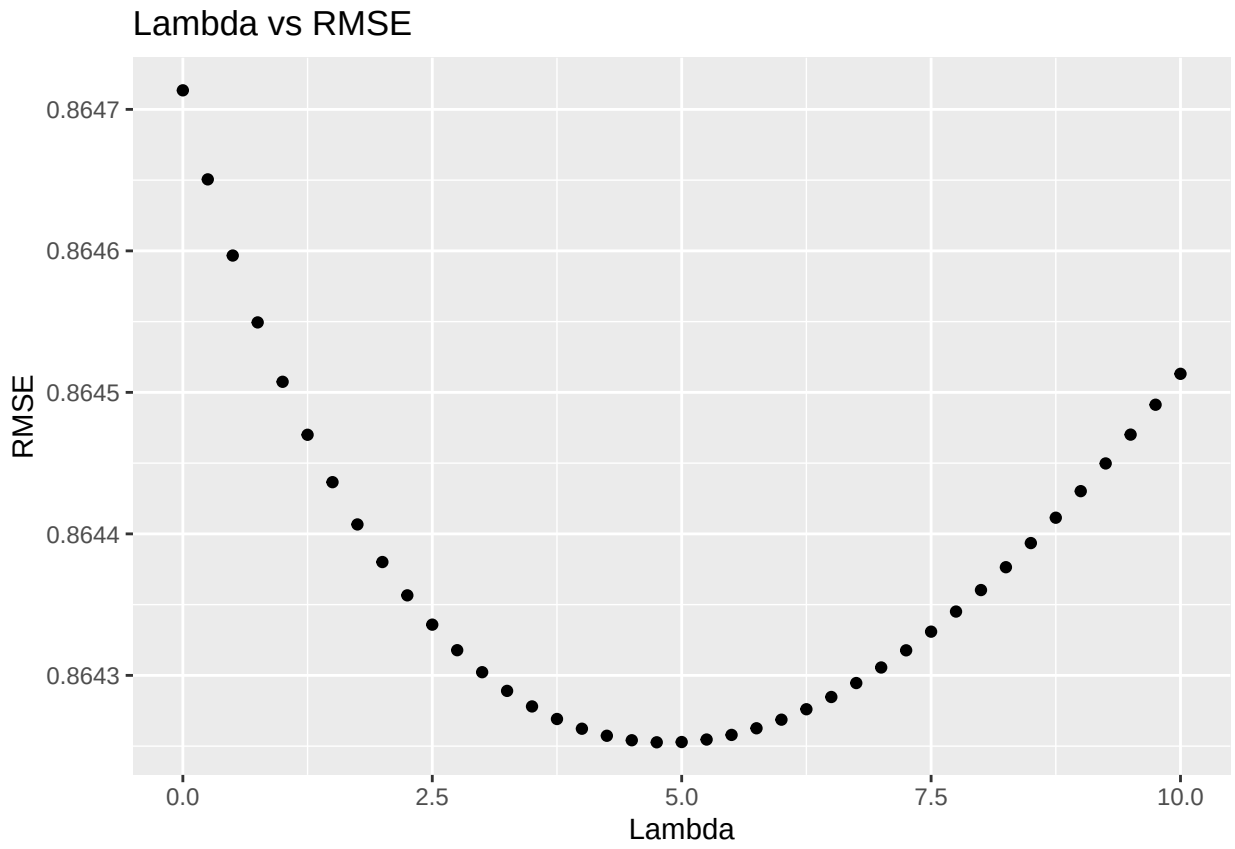
```
######################## Model 5: account for genre effect ######################

# calculate genre effect and store result in table
genre_avgs <-
  edx %>%
  left_join(movie_avgs, by = 'movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  group_by(genres) %>%
  summarize(b_genres = mean(rating - mu - b_movie - b_user))

# predict rating of movies taking into account the above calculated genre effect
predicted_ratings_movie_user_year_genres <-
  validation %>%
  left_join(movie_avgs, by = 'movieId') %>%
  left_join(user_avgs, by = 'userId') %>%
  left_join(year_avgs, by = 'release_year') %>%
  left_join(genre_avgs, by = 'genres') %>%
  mutate(pred = mu + b_movie + b_user + b_year+b_genres) %>%
  .$pred

# calculate RMSE of model
model_movie_user_year_genres_rmse <- RMSE(validation$rating, predicted_ratings_movie_user_year_genres)

# load results to a table summarizing the various models
rmse_results <-
  bind_rows(rmse_results,
            data_frame(method = "Movie+User+Year+genres Effect Model",
                       RMSE = model_movie_user_year_genres_rmse))

# print out all the RMSEs for all the models explored above
rmse_results %>% knitr::kable()
```

| method | RMSE |
|---|---|
| Just the average | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| Movie+User Effect Model | 0.8653488 |
| Movie+User+Year Effect Model | 0.8650043 |
| Movie+User+Year+genres Effect Model | 0.8647872 |

### Model 6: regularized model with effects from user, movie, genre & year

The sixth model regularize and take into account movie, user, release year and genre effect, and further lowers RMSE to 0.8642. The model explores a range of lambdas and ensure to use a lambda value that minimizes RMSE as shown in the lambda vs RMSE plot displayed below.

```r
###### Model 6: regularized model with effects from user, movie, genre & year #######

# create a list of lambdas to explore
lambdas <- seq(0, 10, 0.25)

# calculate RMSEs of model with different lambdas
rmses <- sapply(lambdas, function(l) {
  b_movie_reg <-
    edx %>%
    group_by(movieId) %>%
    summarize(b_movie_reg = sum(rating - mu) / (n() + l))
  b_user_reg <-
    edx %>%
    left_join(b_movie_reg, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_user_reg = sum(rating - mu - b_movie_reg) / (n() + l))
  b_year_reg <-
    edx %>%
    left_join(b_movie_reg, by = "movieId") %>%
    left_join(b_user_reg, by = "userId") %>%
    group_by(release_year) %>%
    summarize(b_year_reg = sum(rating - mu - b_movie_reg - b_user_reg) / (n() + l))
  b_genres_reg <-
    edx %>%
    left_join(b_movie_reg, by = "movieId") %>%
    left_join(b_user_reg, by = "userId") %>%
    left_join(b_year_reg, by = "release_year") %>%
    group_by(genres) %>%
    summarize(b_genres_reg = sum(rating - mu - b_movie_reg - b_user_reg - b_year_reg) / (n() + l))
  predicted_reg_ratings_movie_user_year_genres <-
    validation %>%
    left_join(b_movie_reg, by = "movieId") %>%
    left_join(b_user_reg, by = "userId") %>%
    left_join(b_year_reg, by = "release_year") %>%
    left_join(b_genres_reg, by = "genres") %>%
    mutate(pred = mu + b_movie_reg + b_user_reg + b_year_reg + b_genres_reg) %>%
    .$pred
  model_reg_movie_user_year_genres_rmse <- RMSE(validation$rating, predicted_reg_ratings_movie_user_yea
  return(model_reg_movie_user_year_genres_rmse)
})

# plot lambdas vs RMSEs to graphically display lambdas that lead to minimum RMSE
qplot(lambdas, rmses) + labs(title = "Lambda vs RMSE", x= "Lambda", y = "RMSE")
```




```
# store the best RMSE
lambda_min <- lambdas[which.min(rmses)]
rmses_min <- min(rmses)

# load results to a table summarizing the various models
rmse_results <-
  bind_rows(rmse_results,
            data_frame(method = "Movie+User+Year+genres Regularized Model",
                       RMSE = rmses_min))

# print out all the RMSEs for all the models explored above
rmse_results %>% knitr::kable()
```

| method | RMSE |
|---|---|
| Just the average | 1.0612018 |
| Movie Effect Model | 0.9439087 |
| Movie+User Effect Model | 0.8653488 |
| Movie+User+Year Effect Model | 0.8650043 |
| Movie+User+Year+genres Effect Model | 0.8647872 |
| Movie+User+Year+genres Regularized Model | 0.8642527 |

# Conclusion

The report analysed the movielens data set and explored a variety of increasingly sophisticated ML movie recommendation system that predicts the rating of movies. The best model regularized and took into account movie, user, release year and genre effect, and furnished a model with RMSE of 0.8642 when tested against the validation data set.