Project MovieLens Data Exploration and Analysis

Yoganand Tadepalli

5/14/2020

```
###---------------------------- MovieLens Poject: R Script -----------------------###
##  Compiled in R version 3.6.2(2019-12-12)
##  Scripted in R Studio version 1.2.5033 on Windows 10 operating system


# Excutive Summary and Objective:

# In October 2006, Netflix announced "The Netflix Prize," a competition to make the
# company's recommendation engine 10% more accurate primarily for movies. The dataset
# is over 100 million ratings of 17,770 movies from 480,189 customers.The dataset can
# be downloaded from the following links:
#  [MovieLens 10M dataset] https://grouplens.org/datasets/movielens/10m/
#  [MovieLens 10M dataset - zip file] http://files.grouplens.org/datasets/movielens/ml-10m.zip

# The objective of this capstone project is outlined below:
# 1. Download and prepare the dataset for analysis
# 2. Data exploration and visulization
# 3. Insights into methods used for analysis
# 4. Accuracy using RMSE. Target RMSE < 0.86490
# 5. Conclusion
```

```
# Install and load required libraries for this project. This may take a few minutes.

library(tidyverse)
library(dplyr)
library(ggplot2)
library(data.table)
library(caret)
library(lubridate)
library(tidyr)
library(stringr)
```

```
# Download the MovieLens data files from the above locations

dl <- tempfile()

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


# Dataset preparation. Includes the following steps:
#   Extract Ratings and Movies from the downloaded files (ratings.dat and movies.dat)
#   Add columns for movies
#   Please note ratings is created as data frame and  movies is a matrix. We then
#   transform movies to a data frame and make column data types as appropriate
#   The last step involves creating movielens data frame by combining ratings and
#   transformed movie lens data frame
```

```
#ratings_local <- fread(text = gsub("::", "\t", readLines("C:/Users/Yoganand/projects/MovieLens/ml-10m/
#col.names = c("userId", "movieId", "rating", "timestamp"))


#movies_local <- str_split_fixed(readLines("C:/Users/Yoganand/projects/MovieLens/ml-10m/ml-10M100K/movi


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))


class(movies)
```

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

```
dim(movies)
```

```
## [1] 10681     3
```

```
names(movies)
```

```
## [1] "movieId" "title"   "genres"
```

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


# Explore the above datasets for quick view of statistics

dim(ratings)
```

```
## [1] 10000054        4
```

```
dim(movies)
```

```
## [1] 10681     3
```

```
dim(movielens)
```

```
## [1] 10000054        6
```

```
head(ratings)
```

```
##    userId movieId rating timestamp
## 1:      1     122      5 838985046
## 2:      1     185      5 838983525
## 3:      1     231      5 838983392
## 4:      1     292      5 838983421
## 5:      1     316      5 838983392
## 6:      1     329      5 838983392
```

```
head(movies)
```

```
##   movieId                               title
## 1       1                    Toy Story (1995)
## 2       2                      Jumanji (1995)
## 3       3             Grumpier Old Men (1995)
## 4       4            Waiting to Exhale (1995)
## 5       5 Father of the Bride Part II (1995)
## 6       6                         Heat (1995)
##                                        genres
## 1 Adventure|Animation|Children|Comedy|Fantasy
## 2                  Adventure|Children|Fantasy
## 3                               Comedy|Romance
## 4                        Comedy|Drama|Romance
## 5                                      Comedy
## 6                       Action|Crime|Thriller
```

```
head(movielens)
```

```
##   userId movieId rating timestamp                          title
## 1      1     122      5 838985046                Boomerang (1992)
## 2      1     185      5 838983525                 Net, The (1995)
## 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)
##                          genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 3                        Comedy
## 4  Action|Drama|Sci-Fi|Thriller
## 5        Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
```

```
# Prepare validation set that will be 10% of movielens dataset and ensure same results
# are obtained with setting the seed to 1
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)` instead
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
# Create training and validation sets

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. Finally we remove temporary
# files from the working directory
```

```
removed <- anti_join(temp, validation)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres")
```

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

edx <- edx %>% mutate(year_released = as.numeric(str_extract(str_extract(title, "[/(]\\d{4}[/)]$"), reg

# A quick look at genres column in edx tell us that each movie has multiple genres
# in a single column. We will separate them into columns so we have unique row for
# each movie

# Please note I tried this simple method. However my computer ran out memory with
# several attempts. You may try depending on your computer memory/resources. For now
# it is commented

# genres_split_edx  <- edx  %>% separate_rows(genres, sep = "\\|")

#Here is the alternate method that worked for me.

genrelist <- as.factor(edx$genres)


edx$Action <- ifelse(grepl("Action", edx$genres), 1, 0)
edx$Adventure <- ifelse(grepl("Adventure", edx$genres), 1, 0)
edx$Animation <- ifelse(grepl("Animation", edx$genres), 1, 0)
edx$Children <- ifelse(grepl("Children", edx$genres), 1, 0)
edx$Comedy <- ifelse(grepl("Comedy", edx$genres), 1, 0)
edx$Fantasy <- ifelse(grepl("Fantasy", edx$genres), 1, 0)
edx$"Sci-Fi" <- ifelse(grepl("Sci-Fi", edx$genres), 1, 0)
edx$IMAX <- ifelse(grepl("IMAX", edx$genres), 1, 0)
edx$Drama <- ifelse(grepl("Drama", edx$genres), 1, 0)
edx$Horror <- ifelse(grepl("Horror", edx$genres), 1, 0)
edx$Mystery <- ifelse(grepl("Mystery", edx$genres), 1, 0)
edx$Thriller <- ifelse(grepl("Thriller", edx$genres), 1, 0)
edx$Crime <- ifelse(grepl("Crime", edx$genres), 1, 0)
edx$Romance <- ifelse(grepl("Romance", edx$genres), 1, 0)
edx$War <- ifelse(grepl("War", edx$genres), 1, 0)
edx$Western <- ifelse(grepl("Western", edx$genres), 1, 0)
edx$Music <- ifelse(grepl("Musical", edx$genres), 1, 0)
edx$Documentary <- ifelse(grepl("Documentary", edx$genres), 1, 0)
edx$"Film-Noir" <- ifelse(grepl("Film-Noir", edx$genres), 1, 0)

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


# Now we will look at the above datasets for the dimensions and for any missing values
# This will also check if it is in the tidy format

dim(edx)
```

```
## [1] 9000055      26
```

```
head(edx)
```

```
##   userId movieId rating timestamp                  title
## 1      1     122      5 838985046              Boomerang
## 2      1     185      5 838983525               Net, The
## 3      1     292      5 838983421               Outbreak
## 4      1     316      5 838983392               Stargate
## 5      1     329      5 838983392 Star Trek: Generations
## 6      1     355      5 838984474       Flintstones, The
##                         genres year_released Action Adventure Animation
## 1                Comedy|Romance          1992      0         0         0
## 2         Action|Crime|Thriller          1995      1         0         0
## 3  Action|Drama|Sci-Fi|Thriller          1995      1         0         0
## 4       Action|Adventure|Sci-Fi          1994      1         1         0
## 5 Action|Adventure|Drama|Sci-Fi          1994      1         1         0
## 6       Children|Comedy|Fantasy          1994      0         0         0
##   Children Comedy Fantasy Sci-Fi IMAX Drama Horror Mystery Thriller Crime
## 1        0      1       0      0    0     0      0       0        0     0
## 2        0      0       0      0    0     0      0       0        1     1
## 3        0      0       0      1    0     1      0       0        1     0
## 4        0      0       0      1    0     0      0       0        0     0
## 5        0      0       0      1    0     1      0       0        0     0
## 6        1      1       1      0    0     0      0       0        0     0
##   Romance War Western Music Documentary Film-Noir
## 1       1   0       0     0           0         0
## 2       0   0       0     0           0         0
## 3       0   0       0     0           0         0
## 4       0   0       0     0           0         0
## 5       0   0       0     0           0         0
## 6       0   0       0     0           0         0
```

```
summary(edx)
```

```
##      userId         movieId          rating        timestamp
##  Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
##  1st Qu.:18124   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
##  Median :35738   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35870   Mean   : 4122   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53607   3rd Qu.: 3626   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres          year_released      Action
##  Length:9000055     Length:9000055     Min.   :1915   Min.   :0.0000
##  Class :character   Class :character   1st Qu.:1987   1st Qu.:0.0000
##  Mode  :character   Mode  :character   Median :1994   Median :0.0000
##                                        Mean   :1990   Mean   :0.2845
##                                        3rd Qu.:1998   3rd Qu.:1.0000
##                                        Max.   :2008   Max.   :1.0000
##    Adventure        Animation          Children         Comedy
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.000   1st Qu.:0.0000
##  Median :0.0000   Median :0.00000   Median :0.000   Median :0.0000
##  Mean   :0.2121   Mean   :0.05191   Mean   :0.082   Mean   :0.3934
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.000   Max.   :1.0000
```

```
##     Fantasy           Sci-Fi            IMAX              Drama
##  Min.   :0.0000   Min.   :0.000   Min.   :0.000000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.000000   1st Qu.:0.0000
##  Median :0.0000   Median :0.000   Median :0.000000   Median :0.0000
##  Mean   :0.1028   Mean   :0.149   Mean   :0.000909   Mean   :0.4345
##  3rd Qu.:0.0000   3rd Qu.:0.000   3rd Qu.:0.000000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.000   Max.   :1.000000   Max.   :1.0000
##      Horror           Mystery           Thriller          Crime
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :0.00000   Median :0.00000   Median :0.0000   Median :0.0000
##  Mean   :0.07683   Mean   :0.06315   Mean   :0.2584   Mean   :0.1475
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:0.0000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
##     Romance            War             Western           Music
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.0000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.1902   Mean   :0.05679   Mean   :0.02104   Mean   :0.04812
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
##   Documentary        Film-Noir
##  Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000
##  Mean   :0.01034   Mean   :0.01317
##  3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.00000
```

```r
# Create a summary genre dataframe (initially blank and then fill it using for loop)
# Pick only the individual genre columns we created above


genre_summary <- data.frame(genre = character(), total = numeric())

for (i in 8:26) {
  genre <- colnames(edx)[i]
  total = sum(edx[i] == 1)
  genre_summary <- add_row(genre_summary, genre = genre, total = total)
}

genre_summary
```

```
##          genre   total
## 1       Action 2560545
## 2    Adventure 1908892
## 3    Animation  467168
## 4     Children  737994
## 5       Comedy 3540930
## 6      Fantasy  925637
## 7       Sci-Fi 1341183
## 8         IMAX    8181
## 9        Drama 3910127
## 10      Horror  691485
## 11     Mystery  568332
```

```
## 12    Thriller 2325899
## 13       Crime 1327715
## 14     Romance 1712100
## 15         War  511147
## 16     Western  189394
## 17       Music  433080
## 18 Documentary   93066
## 19   Film-Noir  118541
```

```
# Simple exploration of edx dataset with some charts


edx %>% summarize(
  unique_movies = n_distinct(movieId),
  unique_users = n_distinct(userId))
```

```
##   unique_movies unique_users
## 1         10677        69878
```

```
# Rank the movies in order of number of ratings

edx %>% group_by(movieId, title) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 10,677 x 3
## # Groups:   movieId [10,677]
##    movieId title                                                 count
##      <dbl> <chr>                                                 <int>
##  1     296 "Pulp Fiction "                                       31362
##  2     356 "Forrest Gump "                                       31079
##  3     593 "Silence of the Lambs, The "                          30382
##  4     480 "Jurassic Park "                                      29360
##  5     318 "Shawshank Redemption, The "                          28015
##  6     110 "Braveheart "                                         26212
##  7     457 "Fugitive, The "                                      25998
##  8     589 "Terminator 2: Judgment Day "                         25984
##  9     260 "Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) " 25672
## 10     150 "Apollo 13 "                                          24284
## # ... with 10,667 more rows
```

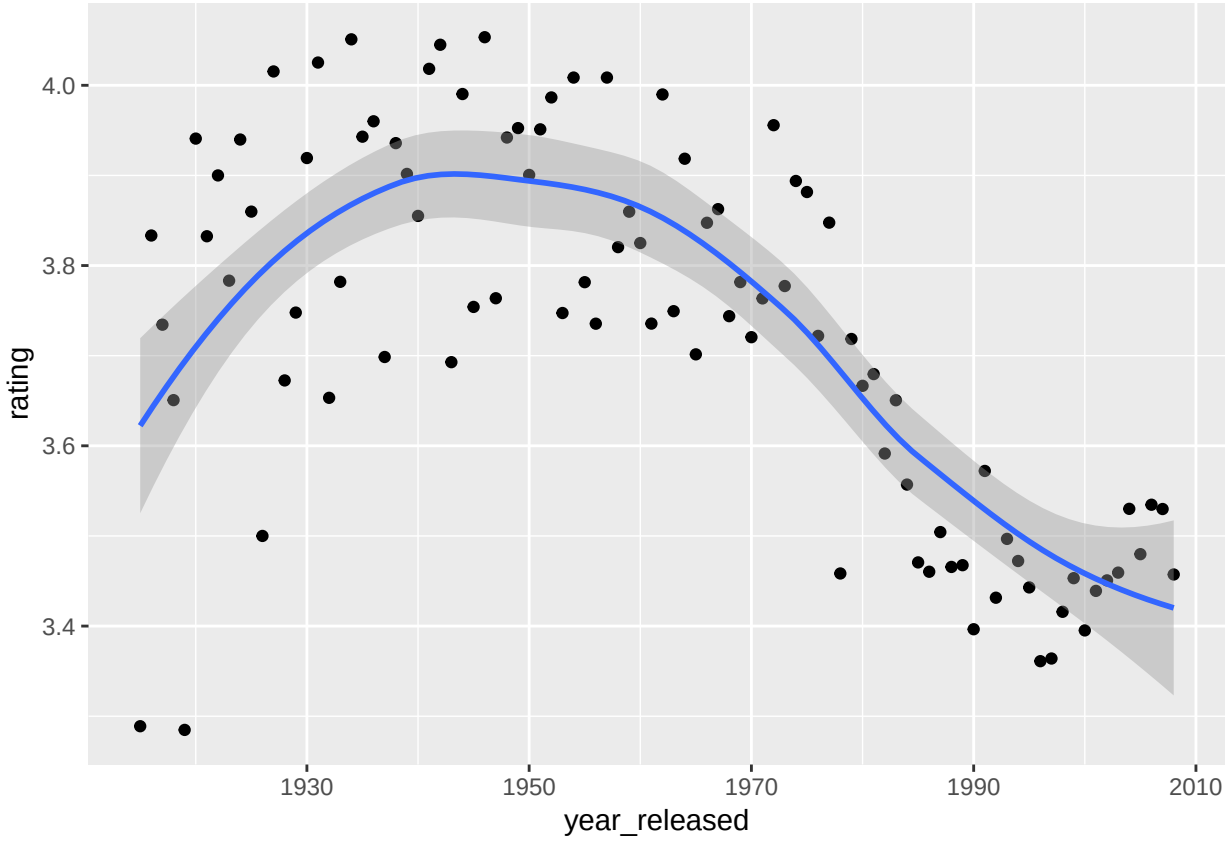
```
# Mean of the rating in edx
mean(edx$rating)
```

```
## [1] 3.512465
```

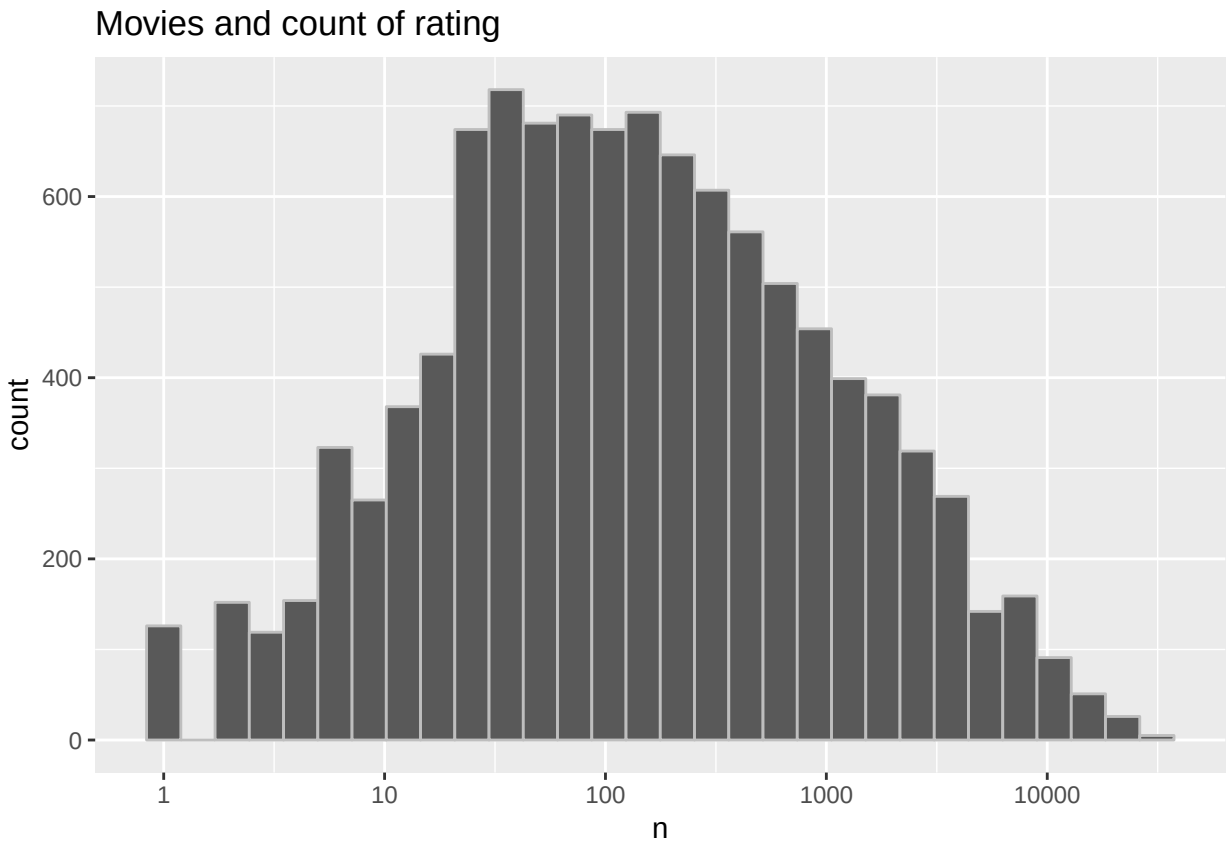
```
# Now we wil review how different features from edx dataset impact each other.
# We will look at ratings, users, movies via different combination of charts.
# Before we start modeling for accuracy, It is important  to get a high-level
# understanding of our data and how it impacts each other.

# Let's start with rating vs year.Note that we have already extracted year as a
# separate column in edx

edx %>% group_by(year_released) %>%
  summarize(rating = mean(rating)) %>%
  ggplot(aes(year_released, rating)) +
```

```
  geom_point() +
  geom_smooth()
```

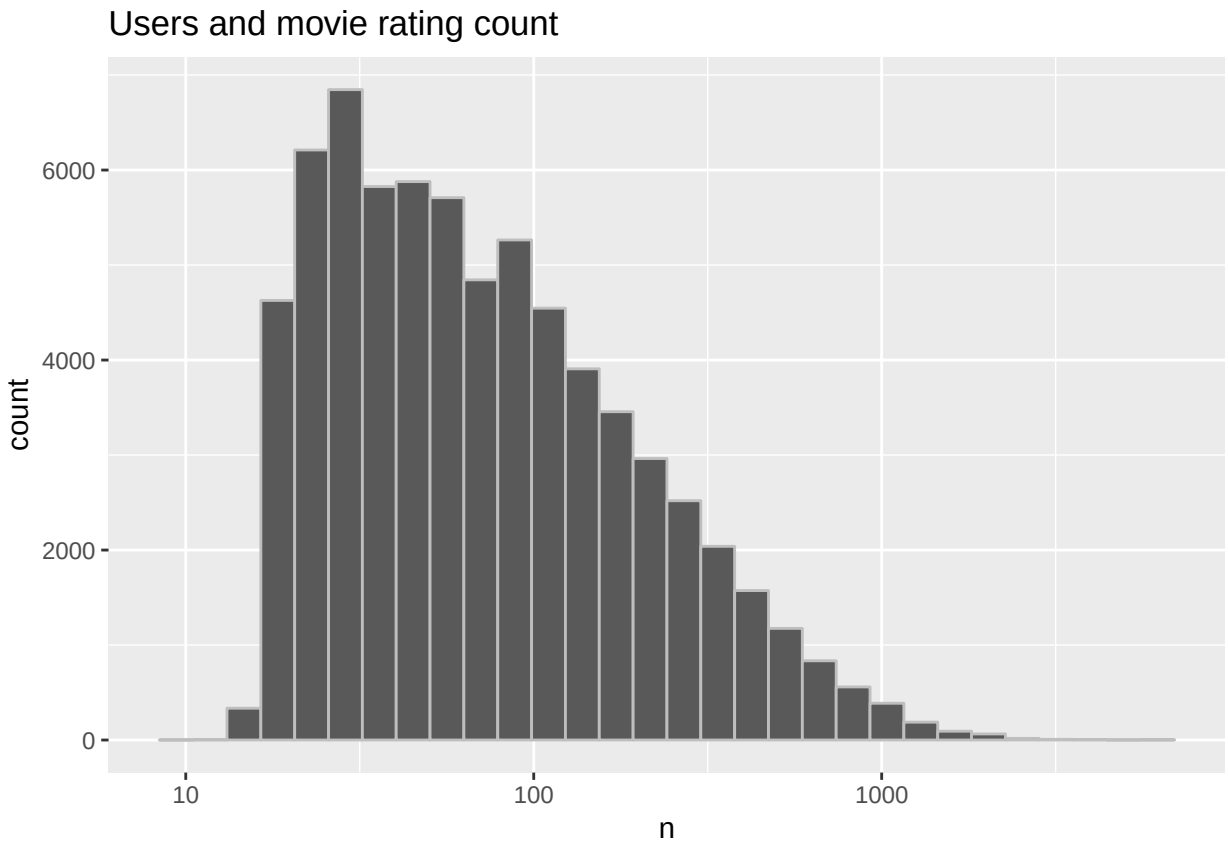
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
# This shows that for newer years fewer ratings

# Code below will plot a chart of movies and rating spread to see how movies
# influencce or biases

edx %>%
  count(movieId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "grey") +
  scale_x_log10() +
  ggtitle("Movies and count of rating ")
```




```
# Here we observe that some movies have been rated signficantly few number of times


# Now lets take a look at users and how they impact movie ratings
edx %>% count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram(bins = 30, color = "grey") +
  scale_x_log10() +
  ggtitle("Users and movie rating count")
```




```
# From the plot, we see there is a potential user bias on the outcome. We will know
# more as we start modeling


### Modeling Approach: Insights into various methods

# We will start with a simple model and use multiple effects/features to improve our
# accuracy. Basically achieve our goal for RMSE < 0.86490. Here is the high-level
# approach. More details are included as we progress and compare each of them

#1. Basic Naive model
#2. Effects of Movies. We will observe if movie bias exists
#3. Effects of Movies and Users. This combination will help us understand any user bias
#4. Regularization of movies and users since these could be noisy estimates that will
#   impact RMSE
#5. Finally, we will obeserve the movies, users and genre combination effect and compare
#   models in the list


#  As a reminder of our objective, we need to calculate RMSE for each model
#  and try to minimize RMSE as much possible. Lower is the RMSE, higher is the accuracy
#  of the prediction algorithm. We will compile a list of the models for final comparison

#  Here is quick recap of RMSE formula:

#   RMSE <- function(true_ratings, predicted_ratings){
```

```
#       sqrt(mean((true_ratings - predicted_ratings)^2))
#       }


# As a first step, we will get average of all ratings
mu_hat <- mean(edx$rating)
mu_hat
```

```
## [1] 3.512465
```

```
# Naive model using only ratings average

naive_avg_rmse <- RMSE(validation$rating, mu_hat)
naive_avg_rmse
```

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

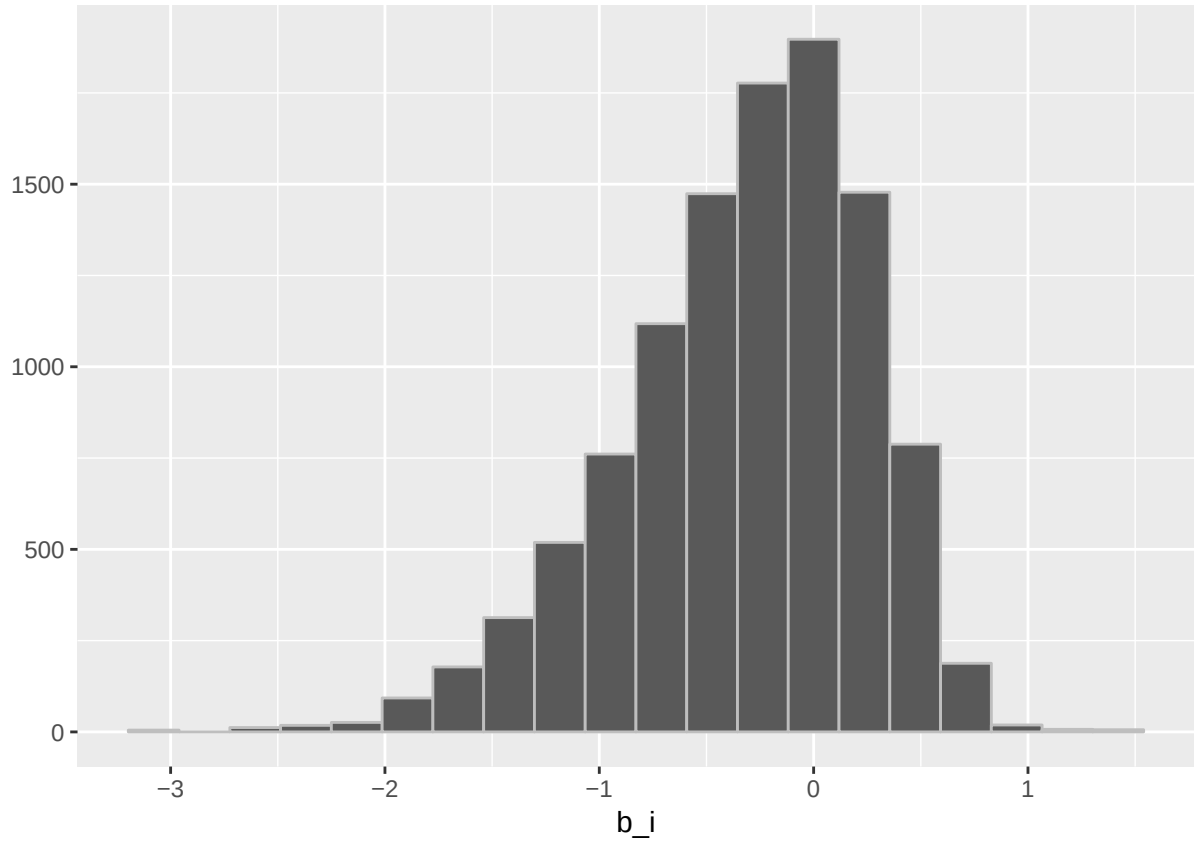
```
# From the ouput, we notice RMSE of 1.061. As we learnt any RMSE value > 1 is not
# an accurate prediction

# Create rmse_results tibble that will store rmse values of all our models  starting
# with just computed Naive RMSE

rmse_results <- tibble(method = "Naive Model", RMSE = naive_avg_rmse)
rmse_results %>% knitr::kable()
```

method	RMSE
Naive Model	1.061202

```
# Second is movies and they are rated. I have picked variable names to
# keep consistent with the chapters, videos and practice teste.This should help refresh
# any part of the concept/codes used earlier.

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

```
## [1] 3.512465
```

```
# The above mu will be used to obtain b_i which we learnt is a bias. Essentially
# the average rating for any given movie i. We will plot a simple histogram and
# observe how far it is from the dataset average mu

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

movavgs %>%
  qplot(b_i, data=., geom="histogram", bins = 20, color = I("grey"))
```

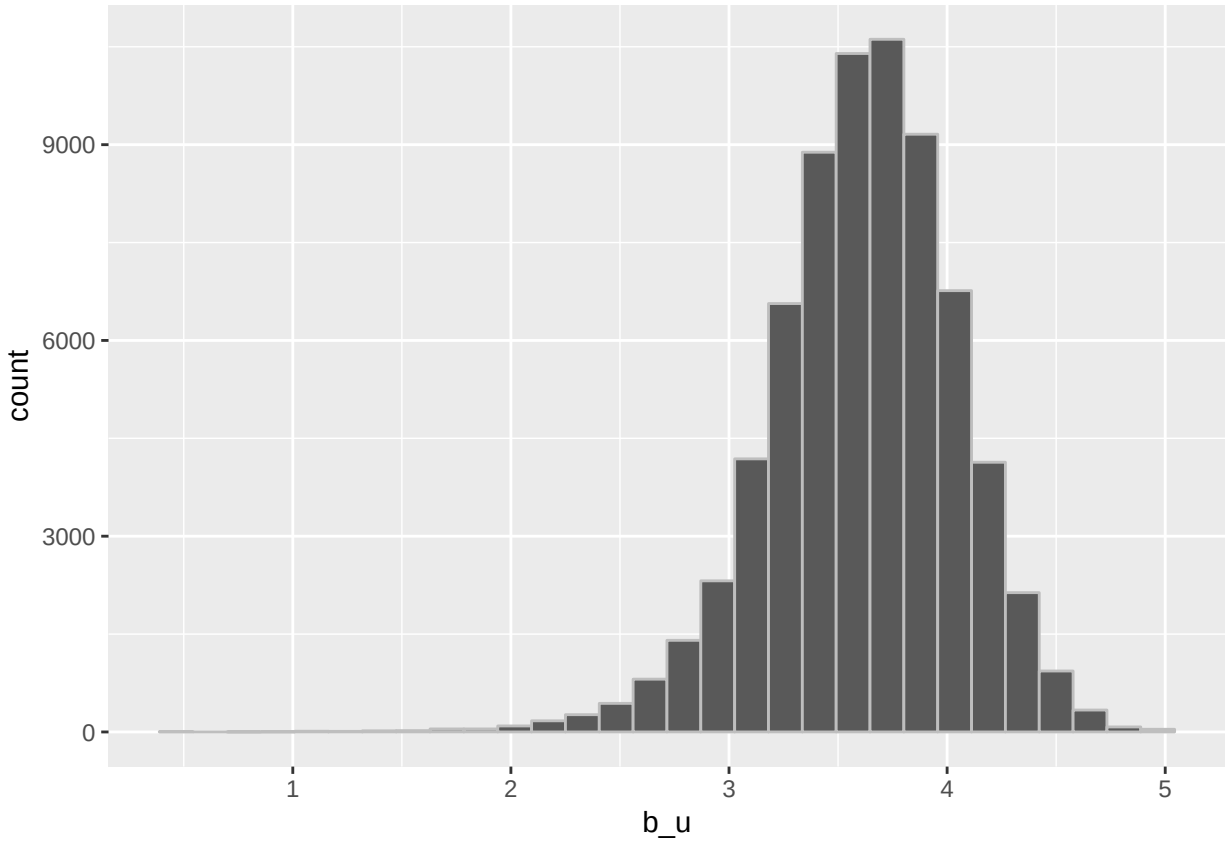



```
# The movies bias is below the dataset average 3.512


# We will now predict with this model
predict_ratings <- mu + validation %>%
  left_join(movavgs, by='movieId') %>%
  .$b_i

movie_rmse <- RMSE(predict_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results, tibble(method = "Movie Effect",
                                               RMSE = movie_rmse))

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

method	RMSE
Naive Model	1.0612018
Movie Effect	0.9439087

```
# Our movie effect model gave a very small improvement compared with Naive.


# Our next model is to look at user effect/bias on the ratings

edx %>%
```

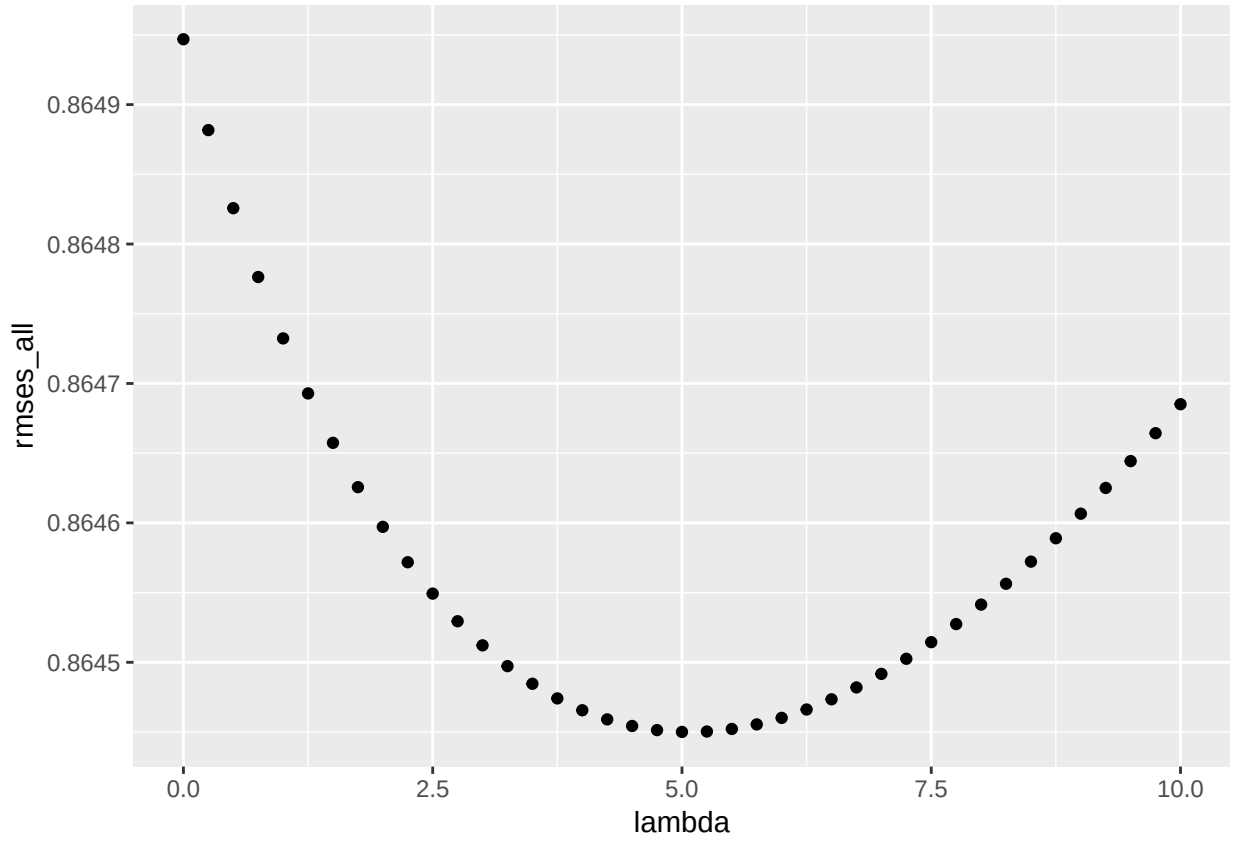
```
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "grey")
```




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

# Predict movie model
predict_ratings <- validation %>%
  left_join(movavgs, by = "movieId") %>%
  left_join(useravgs, by = "userId") %>%
  mutate(pred = mu + b_i + b_u) %>%
  .$pred

user_rmse <- RMSE(predict_ratings, validation$rating)

rmse_results <- bind_rows(rmse_results, tibble(method = "Movie_and_User",
                                               RMSE = user_rmse))

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

method	RMSE
Naive Model	1.0612018
Movie Effect	0.9439087
Movie_and_User	0.8653488

```
# The movie and user combination model gave us accuracy of 0.865. This is a significant
# improvement over the 2nd model. Next we will look at regularizing.

# Reason for regularization: When we take a close look at the movie reviews and how
# they are rated, there is a significant variability. For example, movies with less
# than 5 reviews are rated either very high or very low. This could be noise.
# By regularizing, we attempt to bridge the gap between movies with less ratings
# and dataset average of rating. As we learnt, we need a tunig parameter that will
# give us the most accuracy.

lambda <- seq(0,10,0.25)

rmses_all <- sapply(lambda, function(l){

  mu <- mean(edx$rating)

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

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

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

  predict_ratings <-
    validation %>%
    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

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

# Quick plot where the lambda lies
qplot(lambda, rmses_all, color = I("black"))
```




```
lambda_min <- lambda[which.min(rmses_all)]
lambda_min
```

```
## [1] 5
```

```
# Our tuning parameter that gives us the most accuracy is 5
# We will now summarize results of all rmses and pick the min that gives us the most
# accuracy

rmse_results <- bind_rows(rmse_results,
                          tibble(method="Regularized_Model",
                                 RMSE = min(rmses_all)))

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

method	RMSE
Naive Model	1.0612018
Movie Effect	0.9439087
Movie_and_User	0.8653488
Regularized_Model	0.8644501

```
#From the table above, we have achieved our goal of RMSE < 0.86490
```

```
######## Concluding Comments

# 1. MovieLens dataset was indeed a challenging one to work on. Quite a few attempts
# were made for data preparation and transformation phases

# 2. Regularization of movie, user and genre effects gave us the most accurate
# prediction model

# 3. Please note that to execute the final regularized model, I had to provision
# an AWS instance with higher RAM. My PC had 8GB RAM with 64bit Windows 10, yet
# Rstudio could not execute the model
```