HarvardX MovieLens Capstone Project

Enio Linhares Jr.

2019-04-24

OVERVIEW

In this project, we followed the steps provided during the HarvardX Data Science Course Series (Machine Learning Course Module) in order to find an improvement on the RMSE for recommending movies from the MovieLens Dataset.

The Basic idea is to do an exploratory data analysis and try to select the best features that have more impact when creating an algorithm to recommend a movie.

After an Exploratory Data Analysis through graphical representations and calculating the RMSE's, we have found that the best features for predicting the **ratings** was **movieId** and **userId.**
All the other features (movie age, genre, timestamp, title) did not seem to change the RMSE's final value based on the approaches we have taken considering the hardware limitations.

The final RMSE is 0.8251770

For the full R code of this project please visit this link.

THE DATA SET

Code provided by the edx staff to download an create edx dataset.

Create edx set, validation set, and submission file Note: this process could take a couple of minutes

```
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")

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

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

ratings <- 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(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))

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

# Validation set will be 10% of MovieLens data

set.seed(1)
```

```
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)

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

METHODS AND ANALYSIS

Cleaning your data, exploring and visualizing to find patterns or correlations

```
head(edx)
```

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

We will load some libraries to work with data found

```
library(ggplot2)
library(dplyr)
library(lubridate)
library(tidyr)
```

The data set overview: disctinct movies, genres, and userId's:

```
glimpse(edx)
```

```
## Observations: 9,000,055
## Variables: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
```

```
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 37...
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5...
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 83898339...
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (19...
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|D...
```

```
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
##  Length:9000055     Length:9000055
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

```
n_distinct(edx$movieId)
```

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

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

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

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

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

We notice some differences in the data types that are going to be explored.
The userId feature is basically the user's identification and will be useful when combining data. The movieId feature has the same properties as the userId.
The remaining features will be explored to help us find some patterns in the data to build an efficient algorithm.
This data set has 9000055 observations and 6 variables.
* userId: Unique identification number of each user.
* movieId: Unique identification number of each movie.
* timestamp: Code that contains the date and time in the system format of when the movie was first rated.
* title: Movie title.
* genres: Movie genre (multiple genres per movie are one genre category).
* rating: Rating system for a movie. Ranges from 0 to 5 in 0.5 steps increment.

Exploratory Data Analysis

We converted the "timestamp" feature to a more friendly format called year_tstamp

```
edx <- mutate(edx, year_tstamp = year(as_datetime(timestamp)))
head(edx) # The column "year_tstamp" means the year the movie was first rated.
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
```

```
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     292      5 838983421               Outbreak (1995)
## 4      1     316      5 838983392               Stargate (1994)
## 5      1     329      5 838983392 Star Trek: Generations (1994)
## 6      1     355      5 838984474       Flintstones, The (1994)
##                          genres year_tstamp
## 1                Comedy|Romance        1996
## 2          Action|Crime|Thriller        1996
## 3   Action|Drama|Sci-Fi|Thriller        1996
## 4        Action|Adventure|Sci-Fi        1996
## 5 Action|Adventure|Drama|Sci-Fi        1996
## 6       Children|Comedy|Fantasy        1996
```

The "title" feature shows grouped data that can be useful if split:

```
edx_debut_year <- edx %>%
  # trim whitespaces
  mutate(title = str_trim(title)) %>%
  # split title to title_tmp, debut_year
  extract(title, c("title_tmp", "debut_year"), regex = "^(.*) \\(([0-9 \\-]*)\\)$", remove = FALSE) %>%
  # drop title_tmp column and
  select(-title_tmp)
summary(edx_debut_year) # check for consistency
```

```
##      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            debut_year          genres           year_tstamp
##  Length:9000055     Length:9000055     Length:9000055     Min.   :1995
##  Class :character   Class :character   Class :character   1st Qu.:2000
##  Mode  :character   Mode  :character   Mode  :character   Median :2002
##                                                           Mean   :2002
##                                                           3rd Qu.:2005
##                                                           Max.   :2009
```

```
edx_debut_year$debut_year <- as.numeric(edx_debut_year$debut_year) # convert date to numeric
class(edx_debut_year$debut_year) # check the class
```

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

```
edx_debut_year <- edx_debut_year %>% select(-timestamp) # remove unwanted features
summary(edx_debut_year) # check the consistency again
```

```
##      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
##    debut_year      genres           year_tstamp
##  Min.   :1915   Length:9000055     Min.   :1995
##  1st Qu.:1987   Class :character   1st Qu.:2000
```

```
##  Median :1994   Mode  :character   Median :2002
##  Mean   :1990                      Mean   :2002
##  3rd Qu.:1998                      3rd Qu.:2005
##  Max.   :2008                      Max.   :2009
```

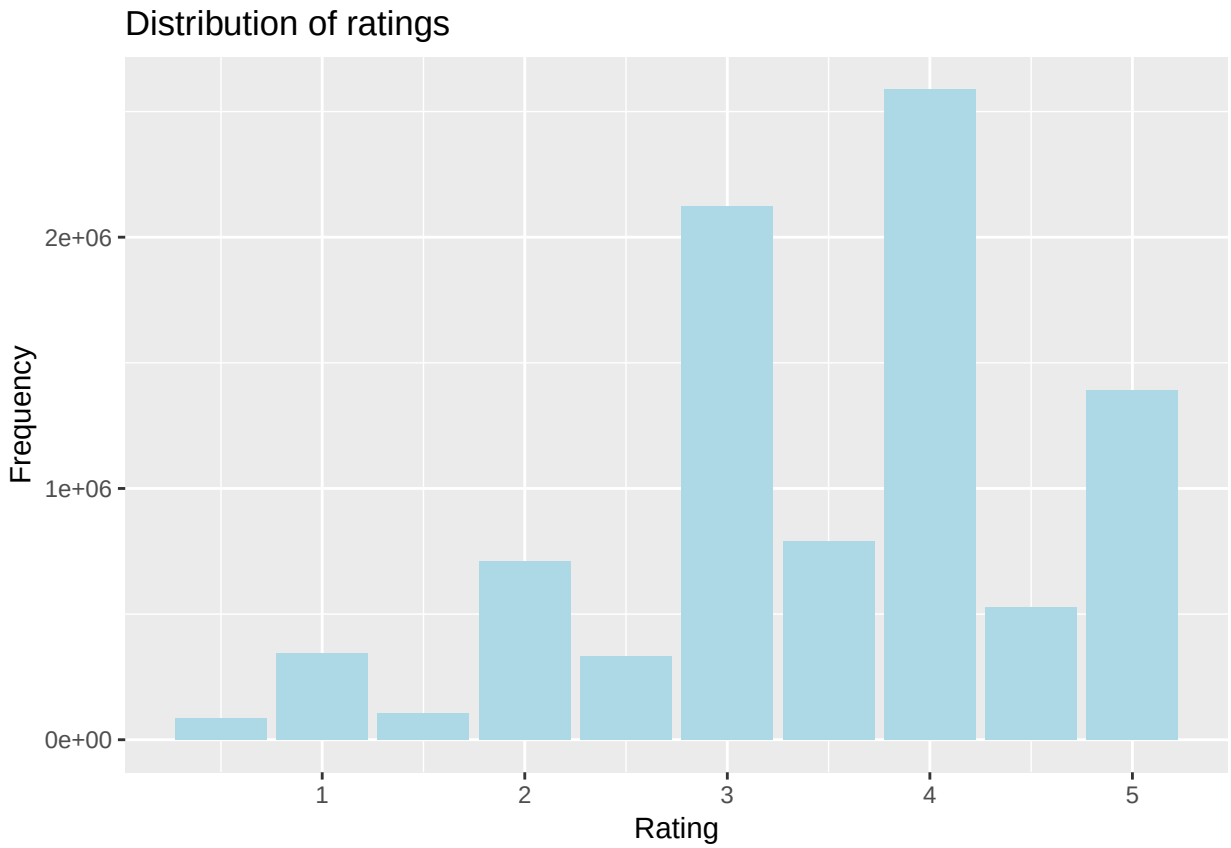
Create a new feature: The individual movies' age

```
edx_debut_year <- edx_debut_year %>% mutate(movie_age = 2019 - debut_year)
edx_clean <- edx_debut_year[, c(1, 2, 3, 4, 6, 5, 7, 8)] # reorder the columns
summary(edx_clean) # check the data for NA's or any irregularity
```

```
##      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            debut_year    year_tstamp     movie_age
##  Length:9000055     Min.   :1915   Min.   :1995   Min.   : 11.00
##  Class :character   1st Qu.:1987   1st Qu.:2000   1st Qu.: 21.00
##  Mode  :character   Median :1994   Median :2002   Median : 25.00
##                     Mean   :1990   Mean   :2002   Mean   : 28.78
##                     3rd Qu.:1998   3rd Qu.:2005   3rd Qu.: 32.00
##                     Max.   :2008   Max.   :2009   Max.   :104.00
```

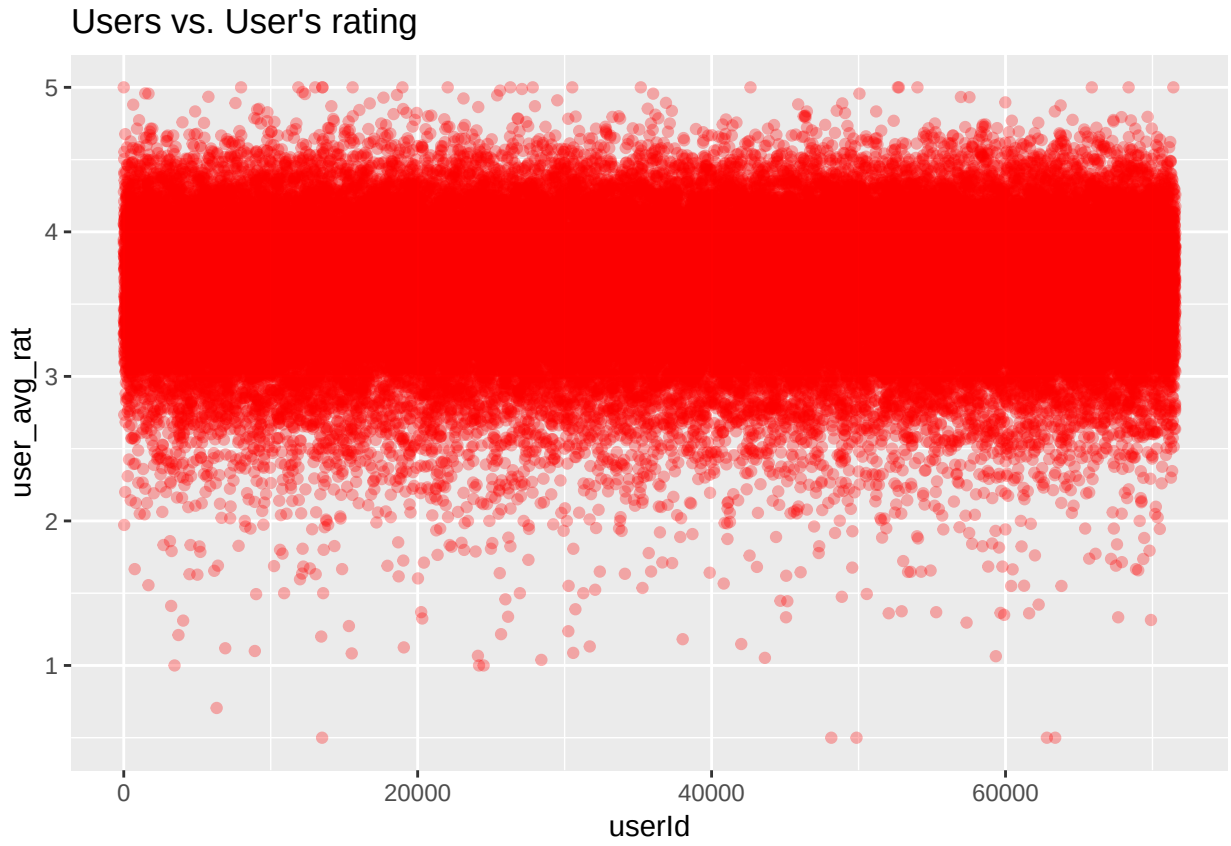
The *rating* feature for movies and users:

```
edx_clean %>%
  group_by(rating) %>%
  summarize(count = n()) %>%
  ggplot(aes(rating, count)) +
  geom_col(fill = "lightblue") +
  labs(title = "Distribution of ratings", x = "Rating",
                y = "Frequency") # checking the frequency of the ratings
```



The users ratings:

```
user_avg_rat <- edx_clean %>% group_by(userId) %>% summarize(user_avg_rat = mean(rating))
ggplot(user_avg_rat, aes(user_avg_rat, userId)) +
  coord_flip() +
  geom_point(alpha = 0.3, colour = "red") +
  ggtitle("Users vs. User's rating")
```




```
summary(user_avg_rat) # the relationship between the usersId and the average rating of each user
```

```
##      userId       user_avg_rat
##  Min.   :    1   Min.   :0.500
##  1st Qu.:17943   1st Qu.:3.357
##  Median :35798   Median :3.635
##  Mean   :35782   Mean   :3.614
##  3rd Qu.:53620   3rd Qu.:3.903
##  Max.   :71567   Max.   :5.000
```

We notice here an even distribution between 3.0 and 4.5

Looking at the movies we have:
1 - A ranking in function of the rating count:

```
most_rated <- edx_clean %>%
  group_by(title) %>% summarise(rating_n =n()) %>% arrange(desc(rating_n))
head(most_rated,20)
```

```
## # A tibble: 20 x 2
##    title                                                     rating_n
##    <chr>                                                        <int>
##  1 Pulp Fiction (1994)                                          31362
##  2 Forrest Gump (1994)                                          31079
##  3 Silence of the Lambs, The (1991)                             30382
##  4 Jurassic Park (1993)                                         29360
##  5 Shawshank Redemption, The (1994)                             28015
##  6 Braveheart (1995)                                            26212
##  7 Fugitive, The (1993)                                         25998
```

```
##  8 Terminator 2: Judgment Day (1991)                                25984
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)     25672
## 10 Apollo 13 (1995)                                                 24284
## 11 Batman (1989)                                                    24277
## 12 Toy Story (1995)                                                 23790
## 13 Independence Day (a.k.a. ID4) (1996)                             23449
## 14 Dances with Wolves (1990)                                        23367
## 15 Schindler's List (1993)                                          23193
## 16 True Lies (1994)                                                 22823
## 17 Star Wars: Episode VI - Return of the Jedi (1983)                22584
## 18 12 Monkeys (Twelve Monkeys) (1995)                               21891
## 19 Usual Suspects, The (1995)                                       21648
## 20 Fargo (1996)                                                     21395
```

2 - A ranking in function of the rating avg:

```
most_rated_avg <- edx_clean %>%
  group_by(title) %>% summarise(rating_avg = mean(rating)) %>% arrange(desc(rating_avg))
head(most_rated_avg, 20)
```

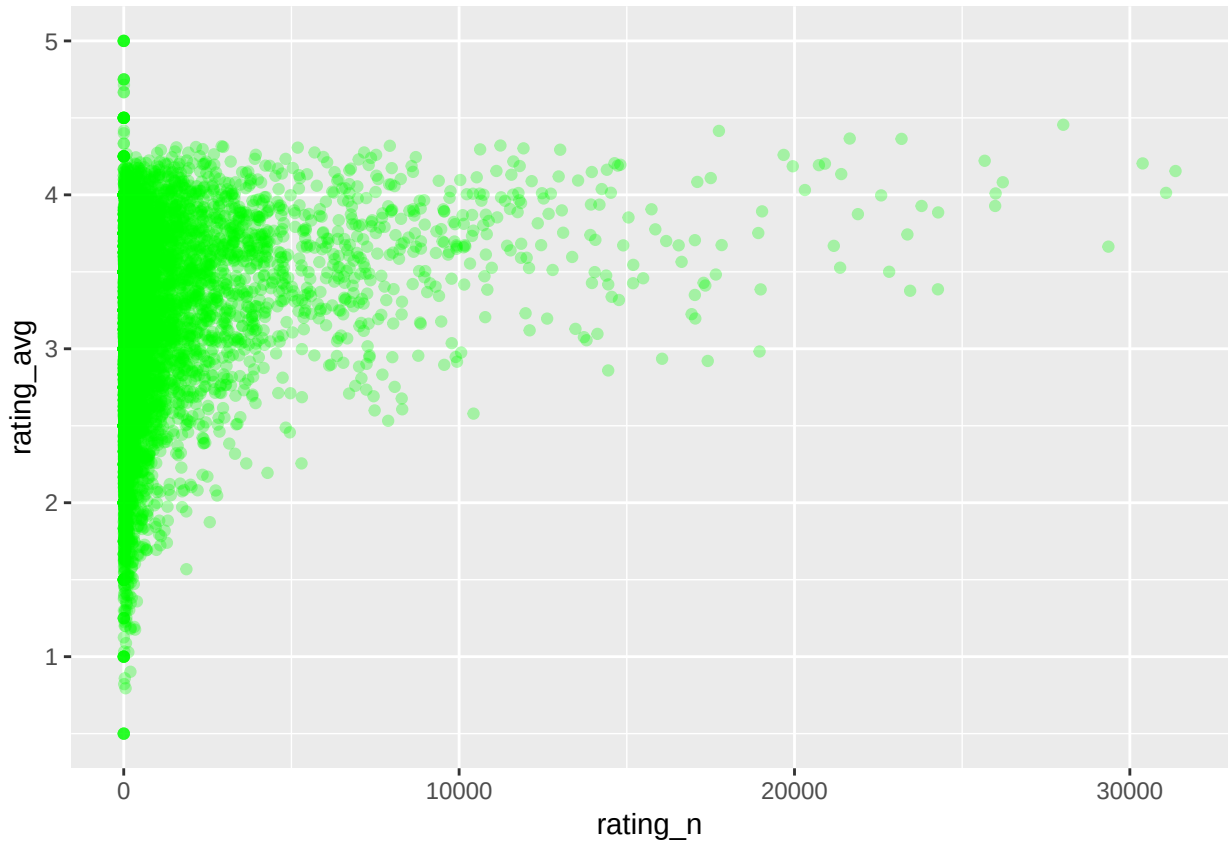
```
## # A tibble: 20 x 2
##    title                                                     rating_avg
##    <chr>                                                          <dbl>
##  1 Blue Light, The (Das Blaue Licht) (1932)                         5
##  2 Fighting Elegy (Kenka erejii) (1966)                             5
##  3 Hellhounds on My Trail (1999)                                    5
##  4 "Satan's Tango (S\u00e1t\u00e1ntang\u00f3) (1994)"               5
##  5 Shadows of Forgotten Ancestors (1964)                            5
##  6 Sun Alley (Sonnenallee) (1999)                                   5
##  7 Constantine's Sword (2007)                                       4.75
##  8 Human Condition II, The (Ningen no joken II) (1959)              4.75
##  9 Human Condition III, The (Ningen no joken III) (1961)            4.75
## 10 Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko ~     4.75
## 11 More (1998)                                                      4.71
## 12 Class, The (Entre les Murs) (2008)                               4.67
## 13 I'm Starting From Three (Ricomincio da Tre) (1981)               4.67
## 14 Bad Blood (Mauvais sang) (1986)                                  4.5
## 15 Demon Lover Diary (1980)                                         4.5
## 16 End of Summer, The (Kohayagawa-ke no aki) (1961)                 4.5
## 17 Kansas City Confidential (1952)                                  4.5
## 18 Ladrones (2007)                                                  4.5
## 19 Life of Oharu, The (Saikaku ichidai onna) (1952)                 4.5
## 20 Man Named Pearl, A (2006)                                        4.5
```

There is a clear difference between the number of ratings a movie has received compared to the rating average.

```
most_rated_avg_n <- left_join(most_rated, most_rated_avg, by = "title")
```

On the next plot we notice that most of the movies were rated less than 5,000 times with some movies being rated above rating 4.0 with very few evaluations.

```
ggplot(most_rated_avg_n, aes(rating_n, rating_avg)) + geom_point(alpha = 0.3, col = "green")
```



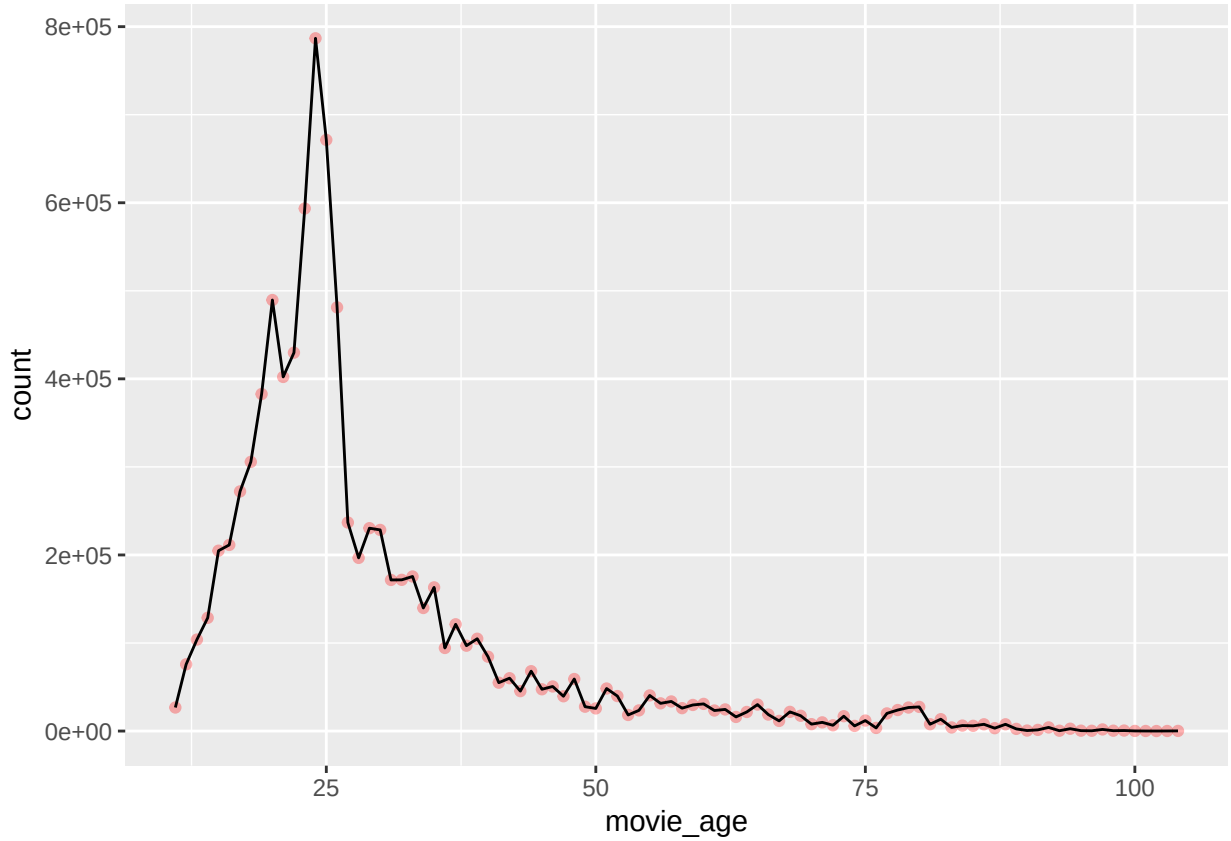
Here we can observe the realtion between the nuber of ratings received with the rating value:

```
head(most_rated_avg_n, 20)
```

```
## # A tibble: 20 x 3
##    title                                                rating_n rating_avg
##    <chr>                                                   <int>      <dbl>
##  1 Pulp Fiction (1994)                                     31362       4.15
##  2 Forrest Gump (1994)                                     31079       4.01
##  3 Silence of the Lambs, The (1991)                        30382       4.20
##  4 Jurassic Park (1993)                                    29360       3.66
##  5 Shawshank Redemption, The (1994)                        28015       4.46
##  6 Braveheart (1995)                                       26212       4.08
##  7 Fugitive, The (1993)                                    25998       4.01
##  8 Terminator 2: Judgment Day (1991)                       25984       3.93
##  9 Star Wars: Episode IV - A New Hope (a.k.a. Star War~    25672       4.22
## 10 Apollo 13 (1995)                                        24284       3.89
## 11 Batman (1989)                                           24277       3.39
## 12 Toy Story (1995)                                        23790       3.93
## 13 Independence Day (a.k.a. ID4) (1996)                    23449       3.38
## 14 Dances with Wolves (1990)                               23367       3.74
## 15 Schindler's List (1993)                                 23193       4.36
## 16 True Lies (1994)                                        22823       3.50
## 17 Star Wars: Episode VI - Return of the Jedi (1983)       22584       4.00
## 18 12 Monkeys (Twelve Monkeys) (1995)                      21891       3.87
## 19 Usual Suspects, The (1995)                              21648       4.37
## 20 Fargo (1996)                                            21395       4.13
```

The age of the movies

```r
movie_age <- edx_clean %>% group_by(movie_age) %>% summarize(count = n())
movie_age %>% ggplot(aes(movie_age, count)) + geom_point(alpha = 0.3, colour = "red") + geom_line()
```




```r
summary(lm(count ~ movie_age, data = movie_age)) # a summary from the linear regression of frequency of
```

```
##
## Call:
## lm(formula = count ~ movie_age, data = movie_age)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -242685  -64452  -22026   29528  565892
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 310512.3    28477.5  10.904  < 2e-16 ***
## movie_age    -3735.1      447.9  -8.339 7.06e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 117800 on 92 degrees of freedom
## Multiple R-squared:  0.4305, Adjusted R-squared:  0.4243
## F-statistic: 69.54 on 1 and 92 DF,  p-value: 7.065e-13
```

There seems to be a split in the plot for movies younger than 25 yo and older, let's check:

```r
movie_age_split_less25 <- edx_clean %>% filter(movie_age <= 25) %>%
  group_by(movie_age) %>% summarize(count = n())
```

```
summary(lm(count ~ movie_age, data = movie_age_split_less25)) # Now we have something if we consider mo
```

```
##
## Call:
## lm(formula = count ~ movie_age, data = movie_age_split_less25)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -104821  -23872   -5130   21242  154403
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -541076      68192  -7.935 2.45e-06 ***
## movie_age      48893       3684  13.272 6.18e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 61640 on 13 degrees of freedom
## Multiple R-squared:  0.9313, Adjusted R-squared:  0.926
## F-statistic: 176.2 on 1 and 13 DF,  p-value: 6.184e-09
```

The above analysis shows good results of R-squared when we select movies with less than 25 years old. This can be attributed to the high number of votes (ratings) for these movies compared to older movies.

We can check now the movies that are older than 25 years.

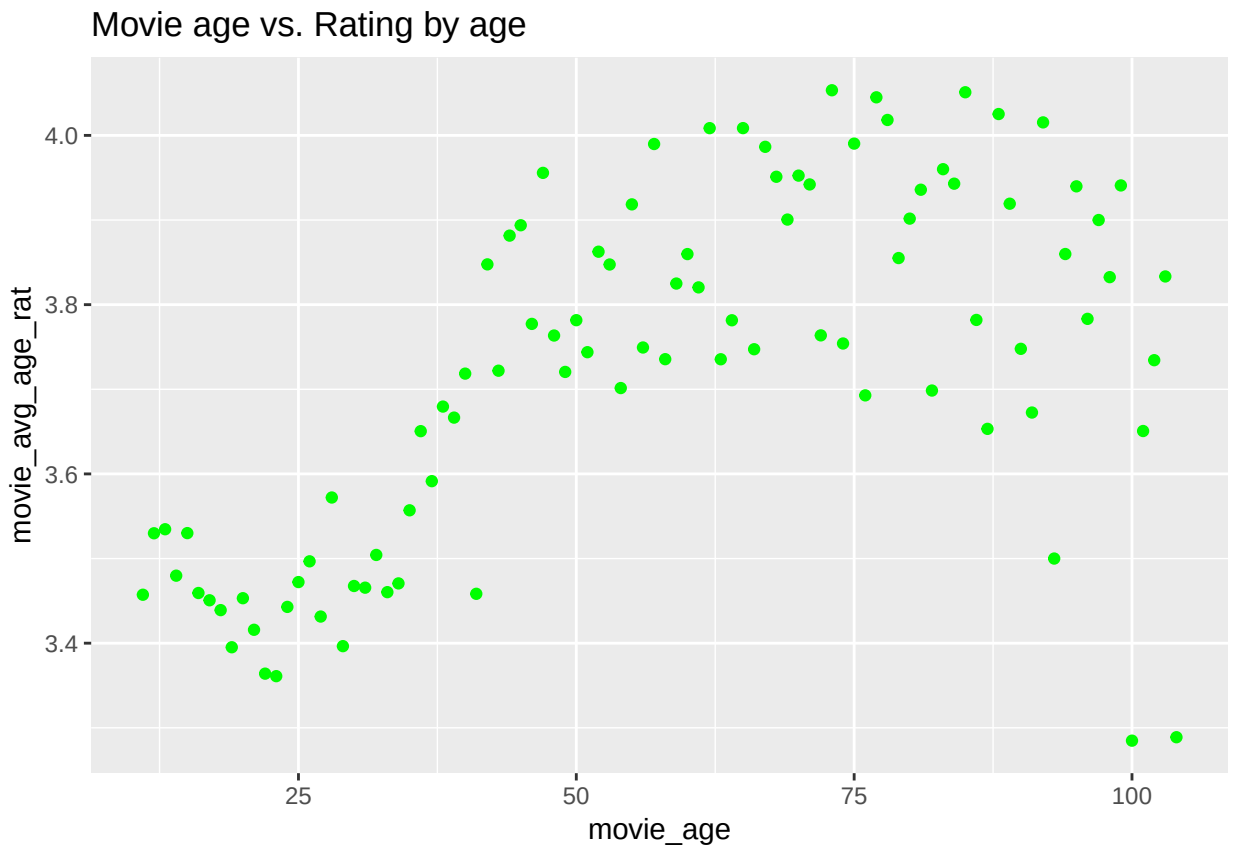
```
movie_age_split_more25 <- edx_clean %>% filter(movie_age > 25) %>%
  group_by(movie_age) %>% summarize(count = n())
summary(lm(count ~ movie_age, data = movie_age_split_more25)) # according to this summary, this time th
```

```
##
## Call:
## lm(formula = count ~ movie_age, data = movie_age_split_more25)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -61115 -33075  -8444  22466 336034
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 208878.8    18213.1   11.47  < 2e-16 ***
## movie_age    -2451.1      264.4   -9.27 3.62e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 53590 on 77 degrees of freedom
## Multiple R-squared:  0.5274, Adjusted R-squared:  0.5213
## F-statistic: 85.94 on 1 and 77 DF,  p-value: 3.617e-14
```

For the Movies let's summarize with some statistics and a plot:

```
# Movie average rating by movieId
movie_avg_rat <- edx_clean %>% group_by(movieId) %>% summarize(movie_avg_rat = mean(rating))
# Movie average rating by time stamp
movie_avg_yr_rat <- edx_clean %>% group_by(year_tstamp) %>% summarize(movie_avg_yr_rat = mean(rating))
# Movie average rating by movie age
```

```r
movie_avg_age_rat <- edx_clean %>% group_by(movie_age) %>% summarize(movie_avg_age_rat = mean(rating))
# a plot about the above data
movie_avg_age_rat %>%
  ggplot(aes(movie_age, movie_avg_age_rat)) +
  geom_point(colour = "Green") +
  ggtitle("Movie age vs. Rating by age")
```




```r
summary(lm(movie_avg_age_rat ~ movie_age, data = movie_avg_age_rat)) # small R-square (0.34)
```

```
##
## Call:
## lm(formula = movie_avg_age_rat ~ movie_age, data = movie_avg_age_rat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.64099 -0.10479  0.01224  0.12053  0.28143
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.4635727  0.0414505  83.559  < 2e-16 ***
## movie_age   0.0044837  0.0006519   6.878  7.2e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1715 on 92 degrees of freedom
## Multiple R-squared:  0.3396, Adjusted R-squared:  0.3324
## F-statistic:  47.3 on 1 and 92 DF,  p-value: 7.198e-10
```

Exploring a little bit more to understand better the above plot:

```
movie_avg_age_rat$movie_age[which.max(movie_avg_age_rat$movie_avg_age_rat)]
```

```
## [1] 73
```

The peak age is 73. We notice that the graph is almost linear between the maximum of 73yo and 25yo.

```
movie_age_25_73 <- movie_avg_age_rat %>% filter((movie_age >= 25) & (movie_age <= 73))
summary(lm(movie_avg_age_rat ~ movie_age, data = movie_age_25_73))
```

```
##
## Call:
## lm(formula = movie_avg_age_rat ~ movie_age, data = movie_age_25_73)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.217770 -0.073989 -0.008052  0.061372  0.236108
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.227318   0.054392  59.334  < 2e-16 ***
## movie_age   0.010475   0.001067   9.821 5.71e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1056 on 47 degrees of freedom
## Multiple R-squared:  0.6724, Adjusted R-squared:  0.6654
## F-statistic: 96.46 on 1 and 47 DF,  p-value: 5.71e-13
```

It makes difference in the R-Squared value now at 0.67 when we limit the range of movie's ages.
There is a second value around 62yo, let's check it:

```
movie_age_25_62 <- movie_avg_age_rat %>% filter((movie_age >= 25) & (movie_age <= 62))
summary(lm(movie_avg_age_rat ~ movie_age, data = movie_age_25_62)) # little bit more improvement at 0.6
```

```
##
## Call:
## lm(formula = movie_avg_age_rat ~ movie_age, data = movie_age_25_62)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.205312 -0.063461  0.003989  0.055099  0.213367
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.125886   0.065546  47.690  < 2e-16 ***
## movie_age   0.013117   0.001461   8.977 1.02e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09877 on 36 degrees of freedom
## Multiple R-squared:  0.6912, Adjusted R-squared:  0.6827
## F-statistic: 80.59 on 1 and 36 DF,  p-value: 1.022e-10
```

A little bit more improvement at 0.69 now.
We notice some interesting correlations here based on the R-squared values:
If we sort by average rating our ranking will be polluted by movies with low count of reviews.

If we sort by number of times a movie was rated by the users we may have a more reliable rating accuracy but we may have some movies that were rated only a few times and thus not being included in the top lists.

To deal with this issue we can use a weighted average technique that can be fully understood and found here.

Another issue arises when we consider the movie age period and we suggest running different analisys when build the algorithm in order to check if there will be an improvement on the RMSE's values and the decide how keep it as one variable.

Let's work on the rating in function of number of ratings (votes)

```
# R = average for the movie (mean) = (rating)
# v = number of votes for the movie = (votes)
# m = minimum votes required to be listed in the Top 250
# C = the mean vote across the whole report
weighted_rat <- function(R, v, m, C) {
  return (v/(v+m))*R + (m/(v+m))*C
}

movie_avg_rat_wt <- edx_clean %>%
  na.omit() %>%
  select(title, rating, debut_year) %>%
  group_by(title, debut_year) %>%
  summarise(count = n(), mean = mean(rating), min = min(rating), max = max(rating)) %>%
  ungroup() %>%
  arrange(desc(mean)) # creating a the dataset with the columns we need

print(movie_avg_rat_wt)
```

```
## # A tibble: 10,676 x 6
##    title                                debut_year count  mean   min   max
##    <chr>                                     <dbl> <int> <dbl> <dbl> <dbl>
##  1 Blue Light, The (Das Blaue Licht) (1~       1932     1  5      5       5
##  2 Fighting Elegy (Kenka erejii) (1966)        1966     1  5      5       5
##  3 Hellhounds on My Trail (1999)               1999     1  5      5       5
##  4 "Satan's Tango (S\u00e1t\u00e1ntang\~       1994     2  5      5       5
##  5 Shadows of Forgotten Ancestors (1964)       1964     1  5      5       5
##  6 Sun Alley (Sonnenallee) (1999)              1999     1  5      5       5
##  7 Constantine's Sword (2007)                  2007     2  4.75   4.5     5
##  8 Human Condition II, The (Ningen no j~       1959     4  4.75   4.5     5
##  9 Human Condition III, The (Ningen no ~       1961     4  4.75   4.5     5
## 10 Who's Singin' Over There? (a.k.a. Wh~       1980     4  4.75   4       5
## # ... with 10,666 more rows
```

Looking at these these numbers and movies we notice that unknown movies are rated higher than blockbuster movies.
Can we change this?

```
movie_weighted_rat <- movie_avg_rat_wt %>%
  mutate(wr = weighted_rat(mean_wt, count, 500, mean(mean))) %>%
  arrange(desc(wr)) %>%
  select(title, debut_year, count, mean, wr) # applying the function created

print(movie_weighted_rat)
```

```
## # A tibble: 10,676 x 5
##    title                                    debut_year count  mean    wr
##    <chr>                                         <dbl> <int> <dbl> <dbl>
##  1 Pulp Fiction (1994)                            1994 31362  4.15 0.984
##  2 Forrest Gump (1994)                            1994 31079  4.01 0.984
##  3 Silence of the Lambs, The (1991)               1991 30382  4.20 0.984
##  4 Jurassic Park (1993)                           1993 29360  3.66 0.983
##  5 Shawshank Redemption, The (1994)               1994 28015  4.46 0.982
##  6 Braveheart (1995)                              1995 26212  4.08 0.981
##  7 Fugitive, The (1993)                           1993 25998  4.01 0.981
##  8 Terminator 2: Judgment Day (1991)              1991 25984  3.93 0.981
##  9 Star Wars: Episode IV - A New Hope (a.k.a.~    1977 25672  4.22 0.981
## 10 Apollo 13 (1995)                               1995 24284  3.89 0.980
## # ... with 10,666 more rows
```

The results are better now, showing not so weird results like before.
The disadvantage of this weighted ratings method is the low score for movies that have a low voting count.
We can use this parameters to develop our algorithm or find a different way to use the same approach (this will be used later as a *lambda* parameter for the weighted averages correction).

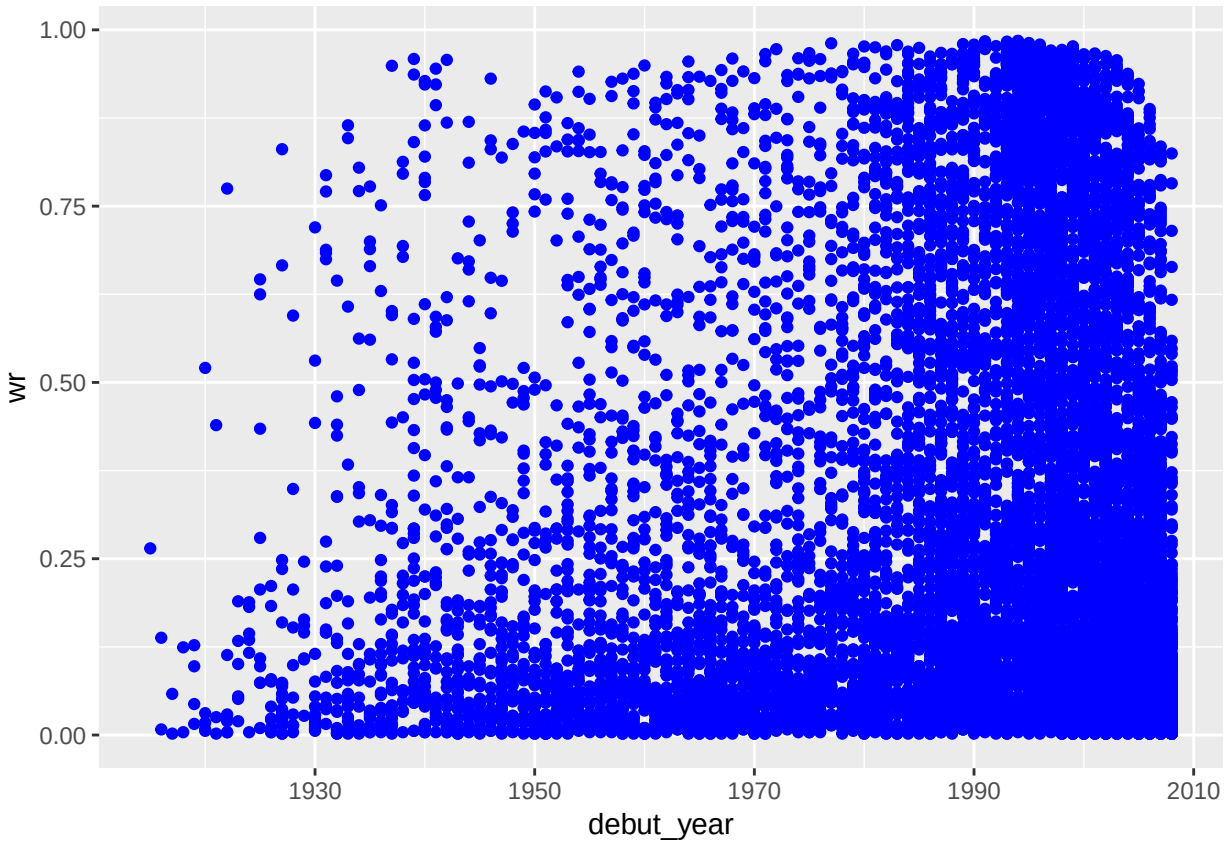
Below we have the results of the movies rated only a few times (sometimes only once):

```
tail(movie_weighted_rat, 20)
```

```
## # A tibble: 20 x 5
##    title                                  debut_year count  mean      wr
##    <chr>                                       <dbl> <int> <dbl>   <dbl>
##  1 Gold Raiders (1951)                          1951     1   2   0.00200
##  2 Moonbase (1998)                              1998     1   2   0.00200
##  3 Rockin' in the Rockies (1945)                1945     1   2   0.00200
##  4 Strange Planet (1999)                        1999     1   2   0.00200
##  5 Won't Anybody Listen? (2000)                 2000     1   2   0.00200
##  6 Adios, Sabata (Indio Black, sai che ti d~    1971     1   1.5 0.00200
##  7 Chapayev (1934)                              1934     1   1.5 0.00200
##  8 Diminished Capacity (2008)                   2008     1   1.5 0.00200
##  9 Flu Bird Horror (2008)                       2008     1   1.5 0.00200
## 10 Hexed (1993)                                 1993     1   1.5 0.00200
## 11 Neil Young: Human Highway (1982)             1982     1   1.5 0.00200
## 12 Dischord (2001)                              2001     1   1   0.00200
## 13 Dog Run (1996)                               1996     1   1   0.00200
## 14 "Monkey's Tale, A (Les Ch\u00e2teau des ~    1999     1   1   0.00200
## 15 Relative Strangers (2006)                    2006     1   1   0.00200
## 16 Stacy's Knights (1982)                       1982     1   1   0.00200
## 17 When Time Ran Out... (a.k.a. The Day the~    1980     1   1   0.00200
## 18 Accused (Anklaget) (2005)                    2005     1   0.5 0.00200
## 19 Confessions of a Superhero (2007)            2007     1   0.5 0.00200
## 20 Hi-Line, The (1999)                          1999     1   0.5 0.00200
```

Here we can visualize the distribution of the corrected ratings in function of the debut year:

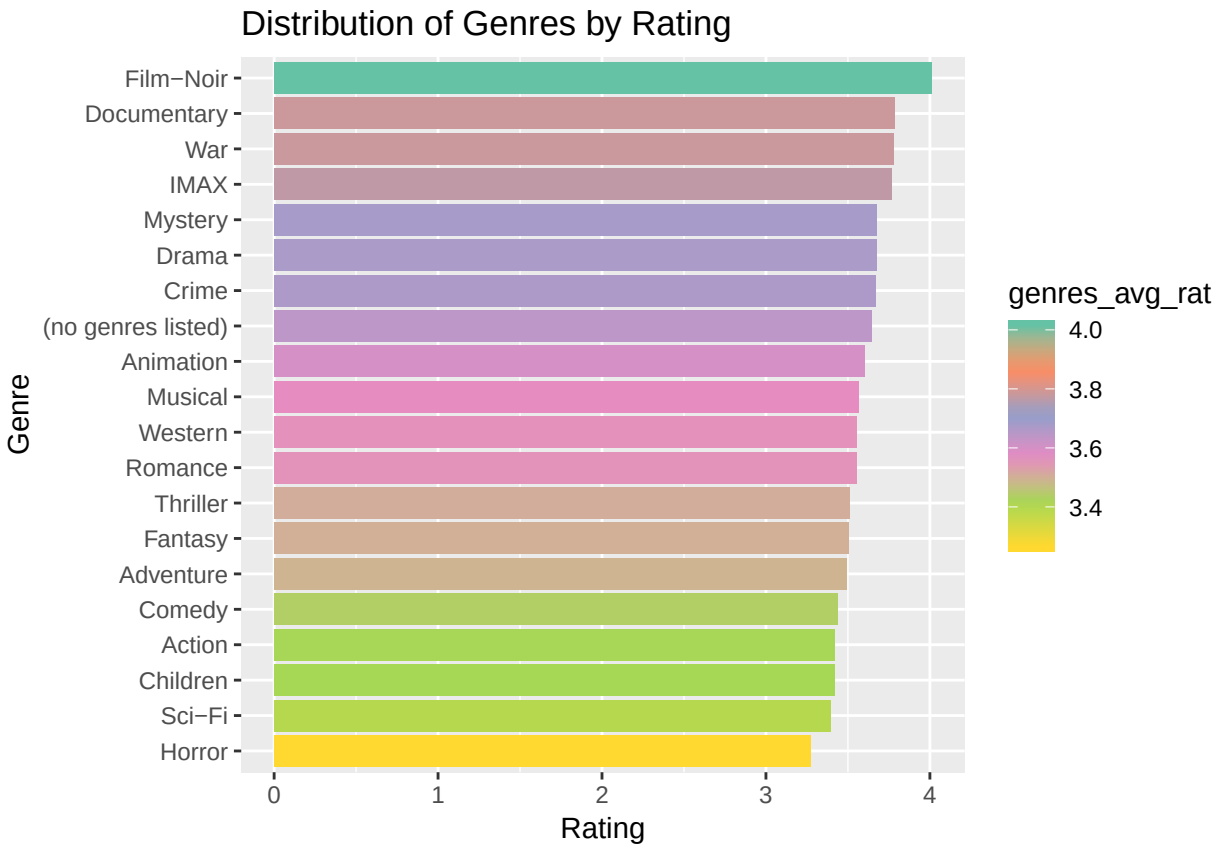
```
ggplot(movie_weighted_rat, aes(x = debut_year, y = wr)) +
  geom_point(col = "blue")
```



What we conclude here is that is better to have this weighted average instead of creating a dummy variable that cuts the rating's value in *more than* and *less than* (i.e rating <=3.0 = 0 & rating > 3.0 = 1).

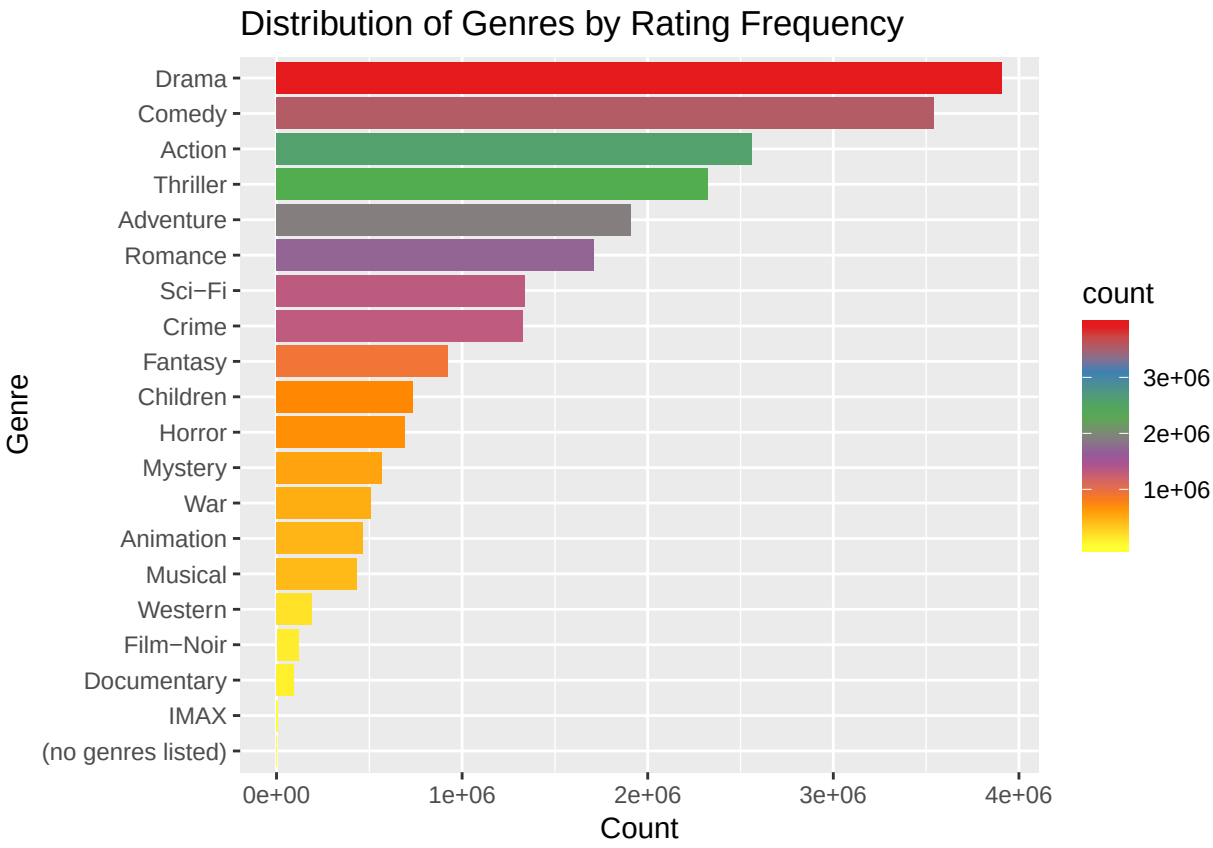
Analysing the genres now:

```
edx_genres <- edx_clean %>% separate_rows(genres, sep = "\\|") # separating the individual genres
genres_avg_rat <- edx_genres %>% group_by(genres) %>% summarize(genres_avg_rat = mean(rating))
ggplot(genres_avg_rat, aes(reorder(genres, genres_avg_rat), genres_avg_rat, fill= genres_avg_rat)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  scale_fill_distiller(palette = "Set2") +
  labs(y = "Rating", x = "Genre") +
  ggtitle("Distribution of Genres by Rating") # after summarizing we visualize
```



On the plot above it does not seem that the rating value changes that much among the genres as the ratings among the genres range from 3.0 to 4.1

```
genres_count <- edx_clean %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarise(count = n()) %>%
  arrange(desc(count))
ggplot(genres_count, aes(reorder(genres, count), count, fill= count)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  scale_fill_distiller(palette = "Set1") +
  labs(y = "Count", x = "Genre") +
  ggtitle("Distribution of Genres by Rating Frequency")
```



Analysing now the by the **"Distribution of Genres by Rating Frequency"** plot of a number of ratings (votes) a genre had we can conclude that some of the genres could play an important role in the algorithm, however being more voted (rated more times) does not mean that a movie is better than another one; This only means that we can rely better on the accuracy of that genre rating than if that genre had been rated fewer times.

Here we have decided to include only a few features in the algorithm development due to some hardware limitations encountered during the computation (these limitations will be discussed at the end of the document).
RESULTS

The Model

The train (edx) and the test (validation) sets have been created with the code provided by the EDX staff and will now be used for the model development.

- The basic model is generated when we consider the average rating from the train set to be predicted into the test set.

```
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
} # defining a formula to calculate the RMSE
```

The rating average:

```
mu_movie_rat <- mean(edx_clean$rating) # calculating the rating average
mu_movie_rat
```

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

The RMSE on the basic model:

```
model_1_rmse <- RMSE(validation$rating, mu_movie_rat) # RMSE in the test set.
model_1_rmse
```

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

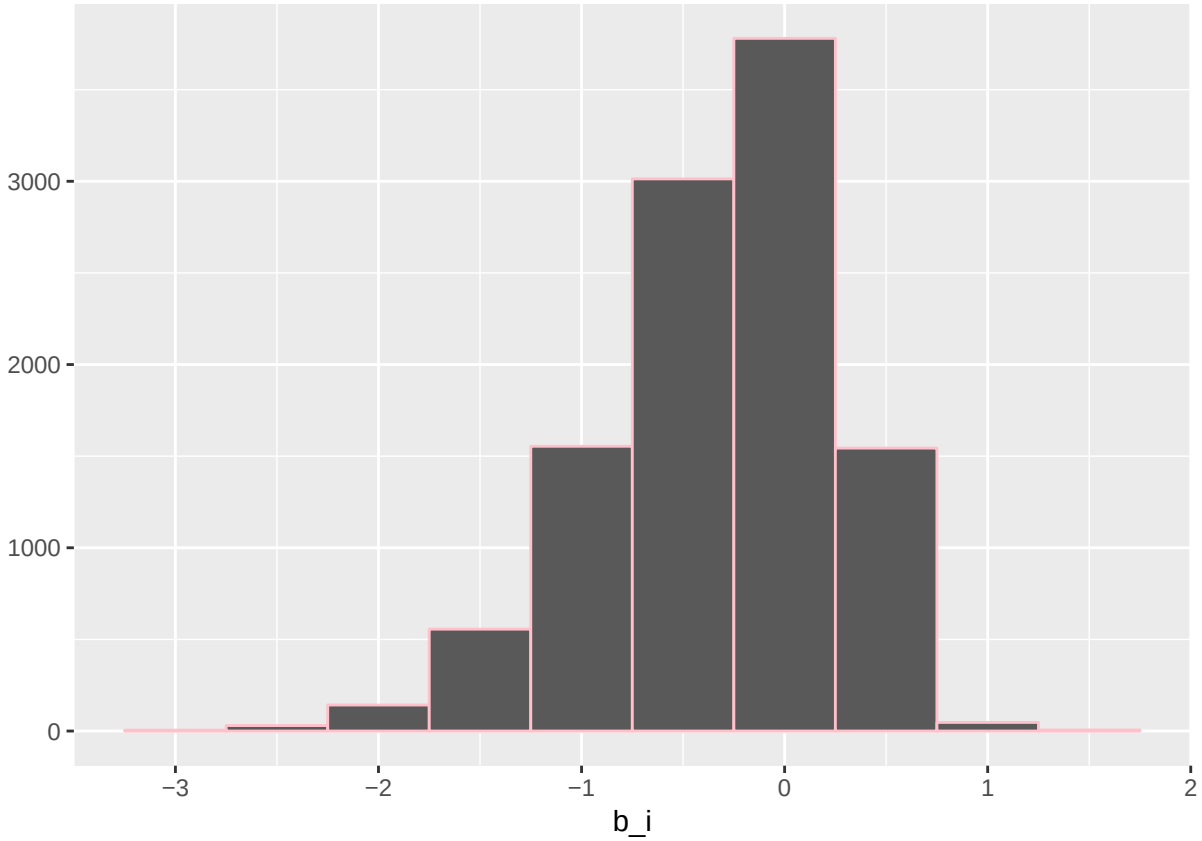
```
rmse_table <- data_frame(Method = "Basic", RMSE = model_1_rmse)
rmse_table %>% knitr::kable(caption = "RMSEs")
```

Table 1: RMSEs

Method	RMSE
Basic	1.061202

- Developing a new model to reduce the RMSE.
  We are considering the movie effect (bi) as a predictor.

```
movie_avgs <- edx_clean %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu_movie_rat))
movie_avgs %>% qplot(b_i, geom ="histogram", bins = 10, data = ., color = I("pink"))
```




```
## Modeling movie effects (b_i)

predicted_ratings <- mu_movie_rat + edx_clean %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)
```

```
model_2_rmse <- RMSE(predicted_ratings, edx_clean$rating)
model_2_rmse
```

```
## [1] 0.9423475
```

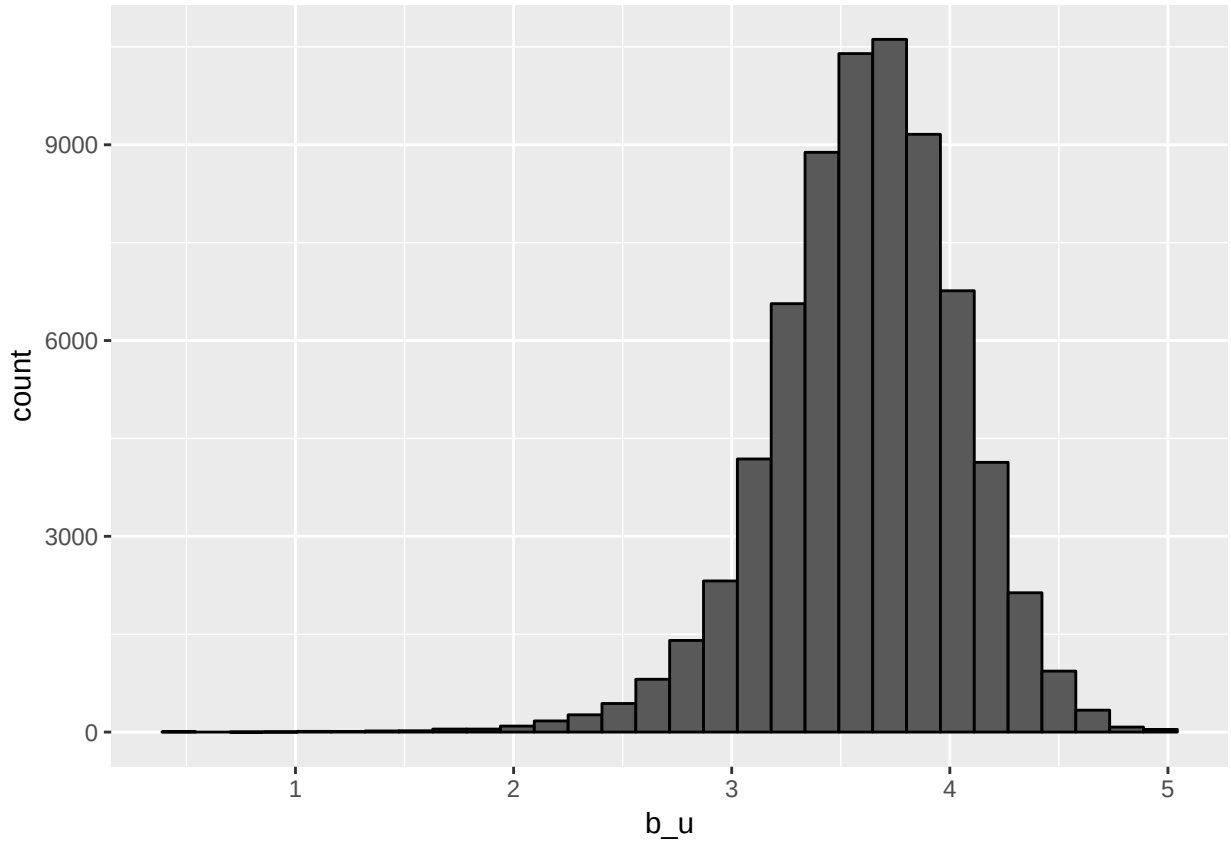
```
rmse_table <- rbind(rmse_table, data_frame(Method = "Movie Effect",
                                           RMSE = model_2_rmse))
rmse_table %>% knitr::kable(caption = "RMSEs")
```

Table 2: RMSEs

Method	RMSE
Basic	1.0612018
Movie Effect	0.9423475

- Developing another model to reduce even more the RMSE.
  We are considering the user effect (bu) as a predictor in combination with the movie effect.

```
edx_clean %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "black")
```




```
user_avgs <- edx_clean %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
```

```
  summarize(b_u = mean(rating - mu_movie_rat - b_i))
# constructing the predictors
predicted_ratings <- edx_clean %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu_movie_rat + b_i + b_u) %>%
  pull(pred)

model_3_rmse <- RMSE(predicted_ratings, edx_clean$rating)
model_3_rmse
```

```
## [1] 0.8567039
```

```
rmse_table <- rbind(rmse_table, data_frame(Method = "Movie Effect + User Effect",
                                           RMSE = model_3_rmse))
rmse_table %>% knitr::kable(caption = "RMSEs")
```

Table 3: RMSEs

Method	RMSE
Basic	1.0612018
Movie Effect	0.9423475
Movie Effect + User Effect	0.8567039

We notice some reduction on the RMSE value. Our model is being improved.

Now we will use this model on the *validation data* (test set):

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

movie_avgs <- validation %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

user_avgs <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))

predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(user_avgs, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

model_val1_rmse <- RMSE(predicted_ratings, validation$rating)
model_val1_rmse
```

```
## [1] 0.825177
```

```
rmse_val_table <- data_frame(Method = "Movie Effect + User Effect on validation",
                                           RMSE = model_val1_rmse)
rmse_val_table %>% knitr::kable(caption = "RMSEs on validation data")
```

Table 4: RMSEs on validation data

Method	RMSE
Movie Effect + User Effect on validation	0.825177

It was noted before that some movies have been rated more than 30,000 times and some movies were rated only 1 time and this causes an imbalance on the rating evaluation and reliability.
We have seen too that using a weighted average on the ratings in function of the number of votes can help us to obtain more precise estimates.
We have decided to use the **Penalized Least Squares** instead of limiting the observations (i.e: creating a subset of the train set with only user that rated more than 30 movies and less than 100). The general idea behind regularization is to constrain the total variability of the effect sizes.

Choosing the Lambda value

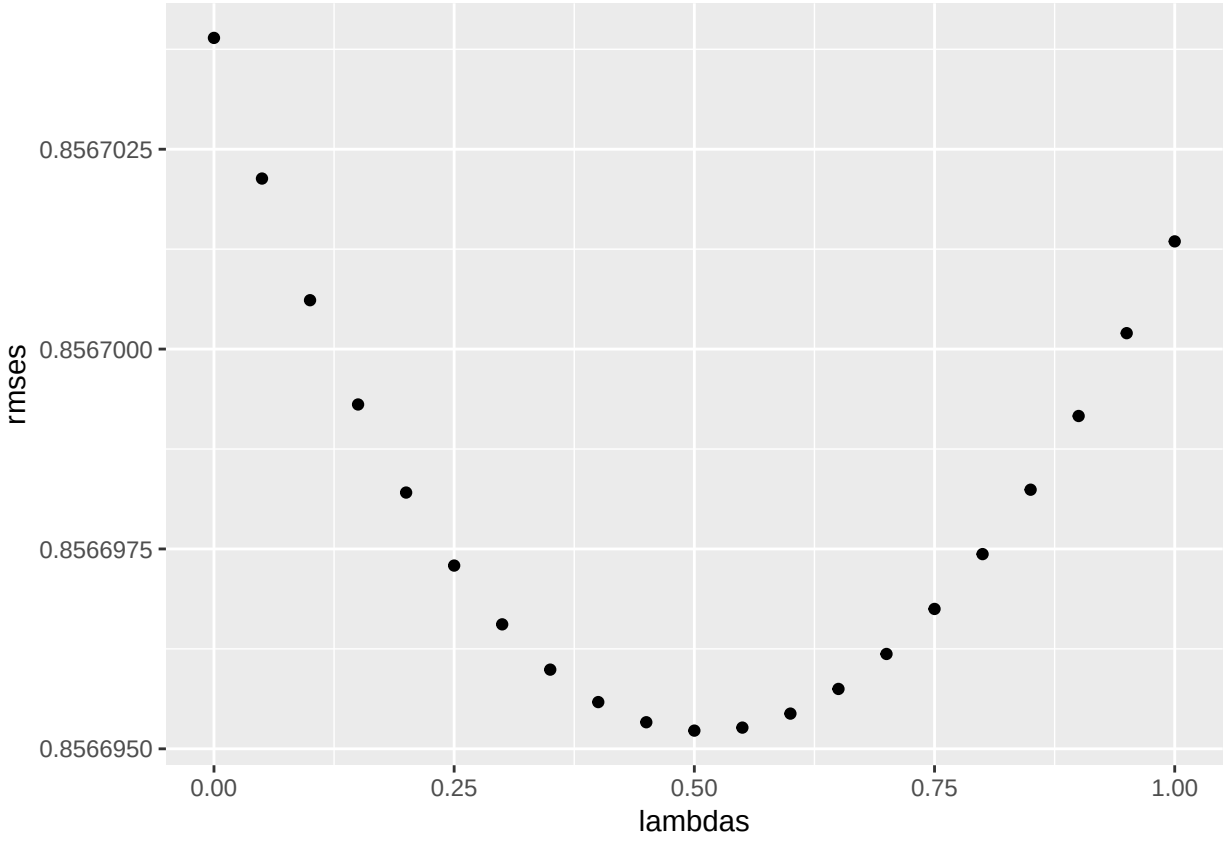
```
lambdas <- seq(0, 1, 0.05)
mu <- mean(edx_clean$rating)
rmses <- sapply(lambdas, function(l){

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

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

  predicted_ratings <-
    edx_clean %>%
    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_clean$rating))
})
qplot(lambdas, rmses)
```




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

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

Using the Penalized Least Squares model and to check the RMSE applying the best *lambda* value

```
mu <- mean(edx_clean$rating)
l <- 0.5
b_i <- edx_clean %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n() + l))

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

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

model_4_rmse <- RMSE(predicted_ratings, edx_clean$rating)
model_4_rmse
```

```
## [1] 0.8566952
```

```
rmse_table <- rbind(rmse_table, data_frame(Method = "Movie Effect + User Effect + Penalized Least Square
                                           RMSE = model_4_rmse))
```

```
rmse_table %>% knitr::kable(caption = "RMSEs")
```

Table 5: RMSEs

Method	RMSE
Basic	1.0612018
Movie Effect	0.9423475
Movie Effect + User Effect	0.8567039
Movie Effect + User Effect + Penalized Least Squares	0.8566952

On the train set (edx_clean) our model performs better than the previous ones.

Using this last model on the Validation data

```
mu <- mean(validation$rating)
l <- 0.5
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

model_val2_rmse <- RMSE(predicted_ratings, validation$rating)
model_val2_rmse
```

```
## [1] 0.8258487
```

```
rmse_val_table <- rbind(rmse_val_table, data_frame(Method = "Movie Effect + User Effect + Penalized Leas
rmse_val_table %>% knitr::kable(caption = "RMSEs on validation data")
```

Table 6: RMSEs on validation data

Method	RMSE
Movie Effect + User Effect on validation	0.8251770
Movie Effect + User Effect + Penalized Least Squares on validation	0.8258487
## RMSE Results	

```
rbind(rmse_table, rmse_val_table) %>% knitr::kable(caption = "RMSEs Summary")
```

Table 7: RMSEs Summary

Method	RMSE
Basic	1.0612018
Movie Effect	0.9423475
Movie Effect + User Effect	0.8567039
Movie Effect + User Effect + Penalized Least Squares	0.8566952
Movie Effect + User Effect on validation	0.8251770
Movie Effect + User Effect + Penalized Least Squares on validation	0.8258487

The better RMSE is achieved with the **Movie Effect + User Effect + Penalized Least Squares** model. However, this RMSE only obtained on the test set.
For model performance reasons we consider what we have obtained from unseeing data on the *validation set (test set)*. The RMSE that has the best performance is the **Movie Effect + User Effect** and will be considered our definitive model. This RMSE is obtained when $\lambda = 0.5$ which lets us to achieve a RMSE equal to 0.8251770.

CONCLUSION

The *userId* and *movieId* variables have sufficient predictive power to permit us predict how a user will rate a movie.
The RMSE equal to 0.8251770 can be considered satisfactory since we have few predictors. Both User and Movie effects carry enough predictive power to forecast the rating that will be given to a movie by a specific user.

Remarks

The main objective of Data science is to collect, clean, and transform data in order to build models that are able to predict as accurate as possible the outcomes of a new and unseen data.

The biggest challenge when we worked on this project was trying to find a balance between the different analytical computational dataset approaches (random forest, gbm, Knn, k-means, clustering), the computational time, and the hardware capacity (home computer).

Unfortunately most of the algorithms could not be used due to the dataset size and time available. This made us make the decision to use a more basic and conservative approach when developing the ML model without loosing sight of the course objectives.