MovieLens_MMC.R

Murilo Mendel Costa

18/08/2020

```
library(tidyverse)
library(caret)
library(data.table)
library(stringr)
library(ggplot2)
library(knitr)

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

# # 1.0 Downloading the data-set
# zip_file <- "~/ml-10m.zip"
# if(file.exists(zip_file)) {dl <- zip_file} else {
#   dl <- tempfile()
#   download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
# }
#
# # Unzip and read files
# 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(movieId),
#                                            title = as.character(title),
#                                            genres = as.character(genres))
#
# movielens <- left_join(ratings, movies, by = "movieId")
#
# # Validation set will be 10% of MovieLens data
# set.seed(1, sample.kind="Rounding")
# test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
# edx <- movielens[-test_index,]
# 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)

load("~/_projects/DataScience/R/HarvardX - DataScience Program/Capstone/data/downloaded.RData")
```

1. Introduction

This document contain a study about MovieLens Dataset, which contains approximately 2,5 billion users ratings about 140 thousand movies over the time. The dataset features available in this study are:

The data set is composed by 9000055 rows, 6 columns and has no missing value. The 6 columns are describe as:

1. userId: User identifier.

2. movieId: Movie identifier.

3. rating: Movie rating given by an user.

4. timestamp (numerical integer value): The time and date in which the rating was provided in seconds since January 1,0 1970

5. title: Movie title.

```
dim(edx)
```

6. genres: Movie genre(s).

```
## [1] 9000055       6
```

```
head(edx)
```

```
##    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
## 1:                Comedy|Romance
```

```
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

```
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
##
##
##
```

We can extract the Number of unique movies and user in the datase.

```
# Number of unique users and movies
edx %>% summarise(unique_Users = n_distinct(userId),
                  unique_Movies = n_distinct(movieId))
```

```
##   unique_Users unique_Movies
## 1        69878         10677
```

Movies with the greatest number of ratings.

```
# Greatest 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 (1994)                                       31362
##  2     356 Forrest Gump (1994)                                       31079
##  3     593 Silence of the Lambs, The (1991)                          30382
##  4     480 Jurassic Park (1993)                                      29360
##  5     318 Shawshank Redemption, The (1994)                          28015
##  6     110 Braveheart (1995)                                         26212
##  7     457 Fugitive, The (1993)                                      25998
##  8     589 Terminator 2: Judgment Day (1991)                         25984
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977) 25672
## 10     150 Apollo 13 (1995)                                          24284
## # ... with 10,667 more rows
```

Top five most given ratings.

```
# Top 5 Most given ratings
edx %>% group_by(rating) %>%
  summarize(count = n()) %>%
  top_n(5) %>%
  arrange(desc(count))
```

```
## # A tibble: 5 x 2
##   rating   count
##    <dbl>   <int>
## 1    4   2588430
## 2    3   2121240
## 3    5   1390114
## 4    3.5  791624
## 5    2    711422
```

Max number of subgenres a movie could have.

```
# Getting the max number of genres in the data_set
edx %>% mutate(genres_quantity = str_count(genres, "\\|") + 1) %>%
  arrange(desc(genres_quantity)) %>%
  select(genres, genres_quantity)
```

```
##                                                                 genres
##       1: Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller
##       2: Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller
##       3: Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller
##       4: Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller
##       5: Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller
##      ---
## 9000051:                                                       Horror
## 9000052:                                                        Drama
## 9000053:                                                        Drama
## 9000054:                                                  Documentary
## 9000055:                                                  Documentary
##          genres_quantity
##       1:               8
##       2:               8
##       3:               8
##       4:               8
##       5:               8
##      ---
## 9000051:               1
## 9000052:               1
## 9000053:               1
## 9000054:               1
## 9000055:               1
```

Conclusion: Movies can have from 0 to 8 sub genres.

Top 30 Best rated movies and the number of votes each of them received

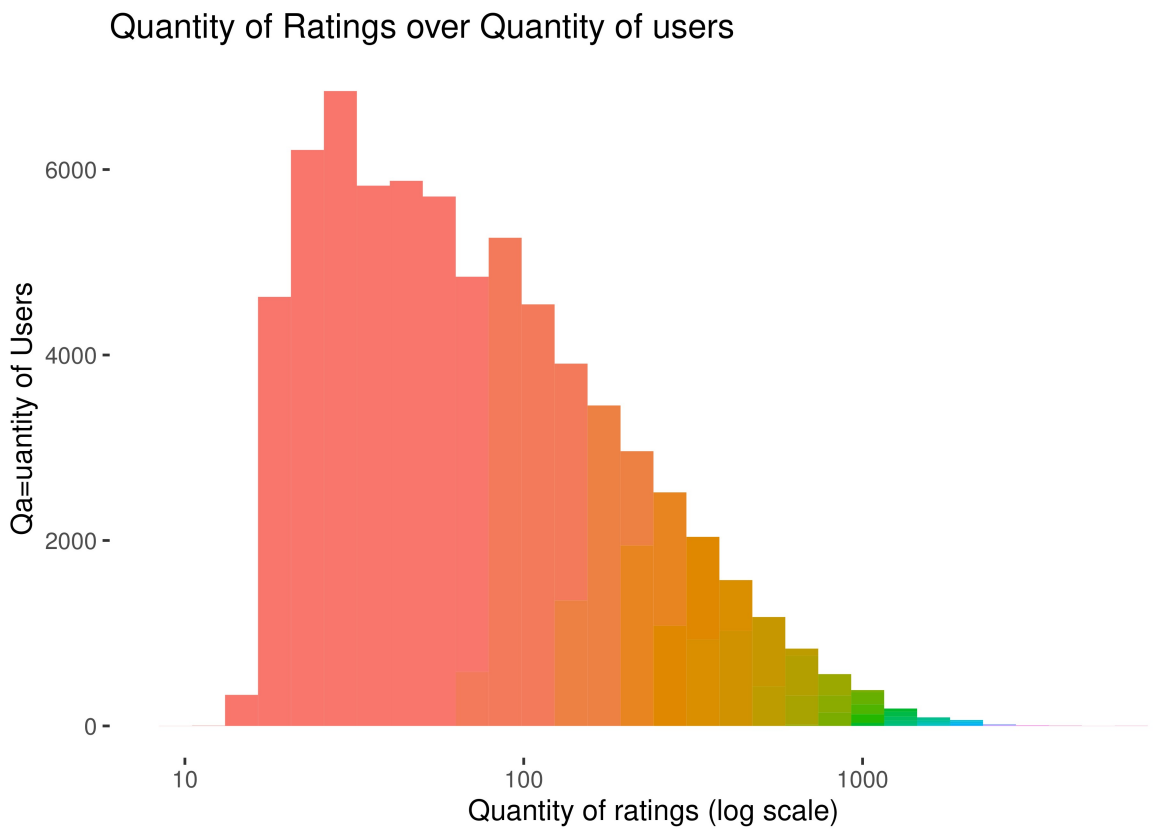
```r
# Top 30 most rated movies and number of votes each of them received
edx %>% group_by(movieId) %>%
  summarise(stars = mean(rating),
            votes = n()) %>%
  left_join(edx, by = "movieId") %>%
  select(movieId, title, votes, stars) %>%
  unique() %>%
  filter(title != "NA") %>%
  arrange(desc(stars)) %>%
  slice(1:30) %>%
  mutate(ranking = 1:n()) %>%
  select(ranking, everything()) %>%
  kable()
```

ranking	movieId	title	votes	stars
1	3226	Hellhounds on My Trail (1999)	1	5.000000
2	33264	Satan's Tango (SÃ¡tÃ¡ntangÃ³) (1994)	2	5.000000
3	42783	Shadows of Forgotten Ancestors (1964)	1	5.000000
4	51209	Fighting Elegy (Kenka erejii) (1966)	1	5.000000
5	53355	Sun Alley (Sonnenallee) (1999)	1	5.000000
6	64275	Blue Light, The (Das Blaue Licht) (1932)	1	5.000000
7	5194	Who's Singin' Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva) (1980)	4	4.750000
8	26048	Human Condition II, The (Ningen no joken II) (1959)	4	4.750000
9	26073	Human Condition III, The (Ningen no joken III) (1961)	4	4.750000
10	65001	Constantine's Sword (2007)	2	4.750000
11	4454	More (1998)	7	4.714286
12	5849	I'm Starting From Three (Ricomincio da Tre) (1981)	3	4.666667
13	63808	Class, The (Entre les Murs) (2008)	3	4.666667
14	7452	Mickey (2003)	1	4.500000
15	7823	Demon Lover Diary (1980)	1	4.500000
16	25975	Life of Oharu, The (Saikaku ichidai onna) (1952)	3	4.500000
17	38320	Valerie and Her Week of Wonders (Valerie a tÃ½den divu) (1970)	1	4.500000
18	50477	Testament of Orpheus, The (Testament d'OrphÃ©e) (1960)	1	4.500000
19	53883	Power of Nightmares: The Rise of the Politics of Fear, The (2004)	4	4.500000
20	56015	Kansas City Confidential (1952)	1	4.500000
21	58185	Please Vote for Me (2007)	1	4.500000
22	60336	Bad Blood (Mauvais sang) (1986)	1	4.500000
23	60990	End of Summer, The (Kohayagawa-ke no aki) (1961)	3	4.500000
24	61695	Ladrones (2007)	1	4.500000
25	63179	Tokyo! (2008)	1	4.500000
26	64418	Man Named Pearl, A (2006)	1	4.500000
27	318	Shawshank Redemption, The (1994)	28015	4.455131
28	32792	Red Desert, The (Deserto rosso, Il) (1964)	6	4.416667
29	858	Godfather, The (1972)	17747	4.415366
30	32657	Man Who Planted Trees, The (Homme qui plantait des arbres, L') (1987)	5	4.400000

Observation: Most of best rated movies has low votes (mostly less than 10)

How many ratings users usually made?

```r
# Quantity of ratings users mostly made
edx %>% count(userId) %>%
  ggplot(aes(n, fill = cut(n, 100))) +
  geom_histogram(bins = 30, show.legend = F) +
  theme_update() +
  labs(x = "Quantity of ratings (log scale)", y = "Qa=uantity of Users") +
  ggtitle("Quantity of Ratings over Quantity of users") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "white")) +
  scale_x_log10()
```

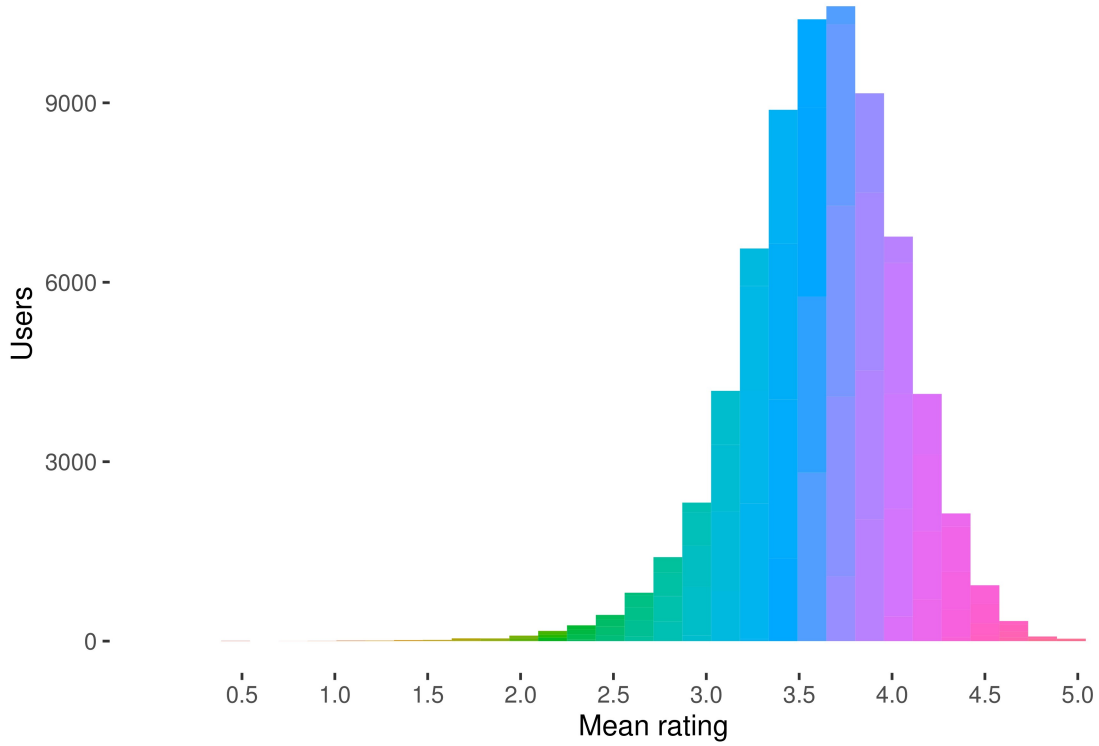


What ratings user usually give?

```r
# Rating users mostly give
edx %>% group_by(userId) %>%
  summarise(m = mean(rating)) %>%
  ggplot(aes(m, fill = cut(m, 100))) +
  geom_histogram(bins = 30, show.legend = F) +
  theme_update() +
  labs(x = "Mean rating", y = "Users") +
  ggtitle("Figure 2.3: Mean rating per users") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "white")) +
  scale_x_discrete(limits = c(seq(min(edx$rating),
```

```
                                    max(edx$rating),
                                    max(edx$rating)/n_distinct(edx$rating))))
```

Figure 2.3: Mean rating per users



2. Data Analysis

2.1 Genres Study

The firs variable which get some attention in the data set is the genres feature. To evaluate if genres feature have correlation with the movie rating, we need to split this character variable by the "|" delimiter.

The first step is discover the max sub-genres a movie in the data set can have.

```
# Getting the max number of sub-genres in the data_set
edx %>% mutate(genres_quantity = str_count(genres, "\\|") + 1) %>%
  arrange(desc(genres_quantity)) %>%
  select(genres, genres_quantity)
```

```
##                                                          genres
##       1: Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller
##       2: Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller
##       3: Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller
##       4: Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller
##       5: Action|Adventure|Comedy|Drama|Fantasy|Horror|Sci-Fi|Thriller
##      ---
```

```
## 9000051:                                                      Horror
## 9000052:                                                       Drama
## 9000053:                                                       Drama
## 9000054:                                                 Documentary
## 9000055:                                                 Documentary
##          genres_quantity
##       1:               8
##       2:               8
##       3:               8
##       4:               8
##       5:               8
##      ---
## 9000051:               1
## 9000052:               1
## 9000053:               1
## 9000054:               1
## 9000055:               1
```

The code above shows:

The most sub-genres a movie in the dataset have are 8; The least sub-genres a movie in the dataset have is 1;

From this observation, we wish to compute the the proportion of each sub-genre in the movies observation.

The "genres" column is separated into 8 new columns, one for each sub-genre identified earlier; It is then calculated the proportion of each sub-genre group in the data set.

```
edx <- edx %>%
  separate(genres, c("genre_1", "genre_2", "genre_3", "genre_4", "genre_5", "genre_6", "genre_7", "genre
  mutate(movie_release = as.integer(str_replace_all(str_extract(title,"\\s\\(\\d{4}\\)"), "\\(|\\)", ""
         title = str_replace(title, "\\s\\(\\d{4}\\)", ""),
         rating_date = as.Date(as.POSIXct(timestamp, origin="1970-01-01")),
         rating_wday = weekdays(rating_date)) %>%
  separate(rating_date, c("rating_year", "rating_month", "rating_mday"))

validation <- validation %>%
  separate(genres, c("genre_1", "genre_2", "genre_3", "genre_4", "genre_5", "genre_6", "genre_7", "genre
  mutate(movie_release = as.integer(str_replace_all(str_extract(title,"\\s\\(\\d{4}\\)"), "\\(|\\)", ""
         title = str_replace(title, "\\s\\(\\d{4}\\)", ""),
         rating_date = as.Date(as.POSIXct(timestamp, origin="1970-01-01")),
         rating_wday = weekdays(rating_date)) %>%
  separate(rating_date, c("rating_year", "rating_month", "rating_mday"))
```

From the code above some observation can be done:

100% of the movies have 1 sub-genre; 80,6% of the movies have 2 sub-genres; 51,3% of the movies have 3 sub-genre; 21,1% of the movies have 4 sub-genres; 5,5% of the movies have 5 sub-genre; 0,8% of the movies have 6 sub-genres; 0,1% of the movies have 7 sub-genre; 0,0028% of the movies have 8 sub-genres;

Conclusions:

We can consider to fit a model with only the first sub-genre (#1); We can consider to fit a moodel with all 8 sub-genres (#2); We can try fill the NA's sub-genres observation by replicating the known genres of each movie (#3).

```
# List of every 16 possible sub-genres
gen <- c("Comedy", "Action", "Children", "Adventure", "Animation", "Drama", "Crime",
         "Sci-Fi", "Horror", "Thriller", "Film-Noir", "Mystery", "Western", "Documentary",
         "Romance", "Fantasy", "Musical", "War", "IMAX", "(no genres listed)")

# Evaluating the mean movie rating for every sub-genre columns over every possible genre
genre_rate <- sapply(gen, function(g){
  r1 <- edx %>% filter(genre_1 == g) %>% summarise(mean_rating_genre_1 = mean(rating))
  r2 <- edx %>% filter(genre_2 == g) %>% summarise(mean_rating_genre_2 = mean(rating))
  r3 <- edx %>% filter(genre_3 == g) %>% summarise(mean_rating_genre_3 = mean(rating))
  r4 <- edx %>% filter(genre_4 == g) %>% summarise(mean_rating_genre_4 = mean(rating))
  r5 <- edx %>% filter(genre_5 == g) %>% summarise(mean_rating_genre_5 = mean(rating))
  r6 <- edx %>% filter(genre_6 == g) %>% summarise(mean_rating_genre_6 = mean(rating))
  r7 <- edx %>% filter(genre_7 == g) %>% summarise(mean_rating_genre_7 = mean(rating))
  r8 <- edx %>% filter(genre_8 == g) %>% summarise(mean_rating_genre_8 = mean(rating))
  as.double(c(r1, r2, r3, r4, r5, r6, r7, r8))
})

# Transpose Matrix
genre_rate <- t(genre_rate)
rowMeans(genre_rate, na.rm = TRUE)
```

```
##             Comedy             Action           Children          Adventure
##           3.534038           3.421405           3.501961           3.505887
##          Animation              Drama              Crime             Sci-Fi
##           3.603006           3.638295           3.535366           3.289522
##             Horror           Thriller          Film-Noir            Mystery
##           3.363272           3.468923           3.985103           3.594321
##            Western        Documentary            Romance            Fantasy
##           3.365542           3.600764           3.567695           3.491574
##            Musical                War               IMAX (no genres listed)
##           3.446324           3.733035           3.340129           3.642857
```

```
# Filling the missing observations over genre_2 to genre_8
# The filling process replicate the existing genres for the movie to the missing genres
edx_filled <- edx %>% mutate(genre_2 = ifelse(is.na(genre_2), genre_1, genre_2),
                genre_3 = ifelse(is.na(genre_3), genre_1, genre_3),
                genre_4 = ifelse(is.na(genre_4), genre_2, genre_4),
                genre_5 = ifelse(is.na(genre_5), genre_1, genre_5),
                genre_6 = ifelse(is.na(genre_6), genre_2, genre_6),
                genre_7 = ifelse(is.na(genre_7), genre_3, genre_7),
                genre_8 = ifelse(is.na(genre_8), genre_4, genre_8))

# mean_rating vs genres matrix
genre_rate_filled <- sapply(gen, function(g){
  r1 <- edx_filled %>% filter(genre_1 == g) %>% summarise(mean_rating_genre_1 = mean(rating))
  r2 <- edx_filled %>% filter(genre_2 == g) %>% summarise(mean_rating_genre_2 = mean(rating))
```

```
  r3 <- edx_filled %>% filter(genre_3 == g) %>% summarise(mean_rating_genre_3 = mean(rating))
  r4 <- edx_filled %>% filter(genre_4 == g) %>% summarise(mean_rating_genre_4 = mean(rating))
  r5 <- edx_filled %>% filter(genre_5 == g) %>% summarise(mean_rating_genre_5 = mean(rating))
  r6 <- edx_filled %>% filter(genre_6 == g) %>% summarise(mean_rating_genre_6 = mean(rating))
  r7 <- edx_filled %>% filter(genre_7 == g) %>% summarise(mean_rating_genre_7 = mean(rating))
  r8 <- edx_filled %>% filter(genre_8 == g) %>% summarise(mean_rating_genre_8 = mean(rating))
  as.double(c(r1, r2, r3, r4, r5, r6, r7, r8))
})

# Transpose Matrix
genre_rate_filled <- t(genre_rate_filled)
rowMeans(genre_rate_filled, na.rm = TRUE)
```

```
##             Comedy             Action           Children          Adventure
##           3.351248           3.143443           3.333564           3.492362
##          Animation              Drama              Crime              Sci-Fi
##           3.595669           3.680787           3.653433           3.359571
##             Horror           Thriller          Film-Noir            Mystery
##           3.175165           3.486115           4.032414           3.696207
##            Western        Documentary            Romance            Fantasy
##           3.571333           3.799927           3.513161           3.483359
##            Musical                War               IMAX (no genres listed)
##           3.544334           3.777187           3.453036           3.642857
```

```
# Mean difference between rates is 9.7% between original model and filled model
mean(abs(rowMeans(genre_rate, na.rm = TRUE) - rowMeans(genre_rate_filled, na.rm = TRUE)))
```

```
## [1] 0.09789429
```

The code above evaluate the mean user rating for each sub-genre, considering both:

> The original model, disregarding the NA's values, and; A model which every NA value is filled by replicating a movie known genre to it.

Conclusions:

1. Filling the missing sub-genres values by replicating a movie known genre implies changing the original mean rates by 9,7% (# Discard Hypothesis #3)

2.2 Movie Release Year Study

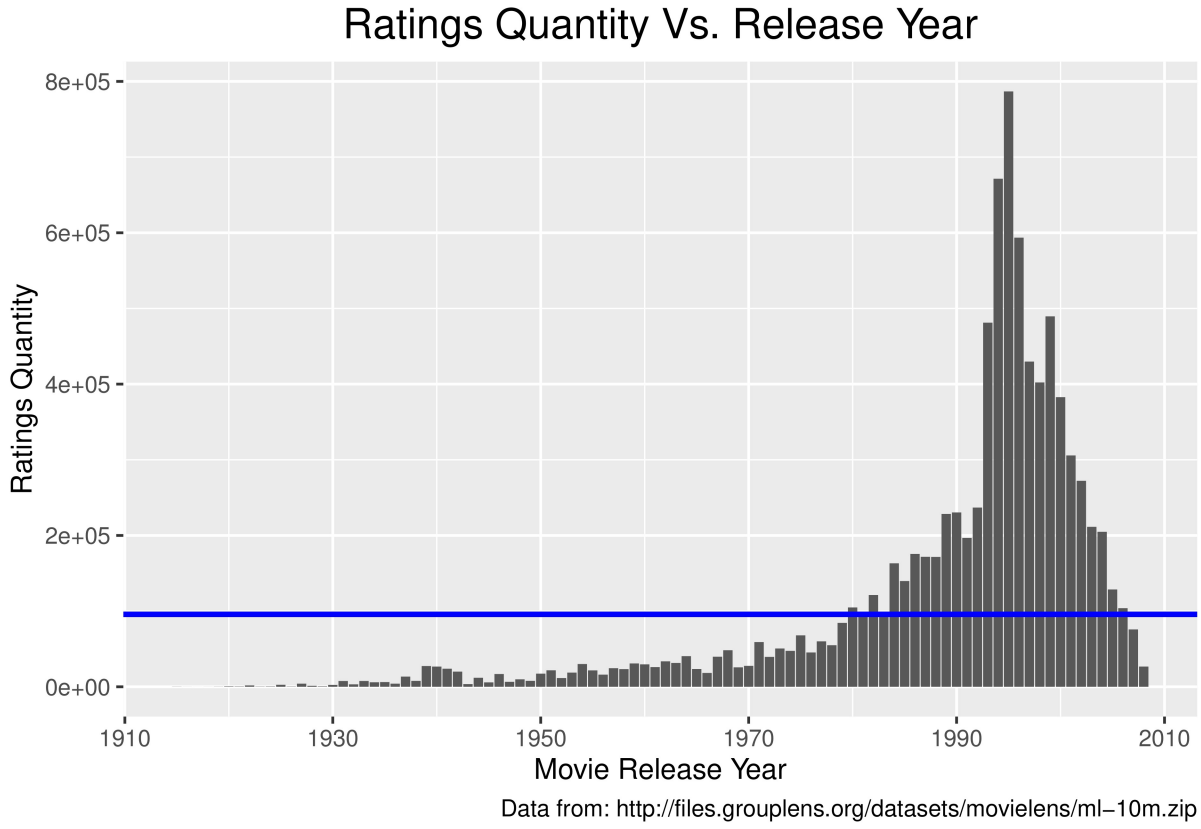
In this section, the ideia is to split the the movie title and extract a new feature about the movie release year.

From this new variable, it's possible to study some correlation between movie ratings for newest and oldest movies.

```
# Negative correlation so close to 0, showing that newest movies in general have lower ratings
cor(edx$rating, edx$movie_release)
```

```
## [1] -0.1207188
```

```
# Plotting Quantity of ratings over movies release year
edx %>% group_by(movie_release) %>%
  summarize(count = n()) %>%
  ggplot() +
  geom_col(aes(movie_release, count)) +
  geom_hline(aes(yintercept = mean(count)), size = 1, colour = 'blue') +
  theme(plot.title = element_text(hjust = 0.5, size = 16)) +
  labs(title = "Ratings Quantity Vs. Release Year",
       caption = "Data from: http://files.grouplens.org/datasets/movielens/ml-10m.zip") +
  xlab("Movie Release Year") +
  ylab("Ratings Quantity")
```



Observations:

The correlation variable between ratings and the movie release year is -0.12, very close to zero, which means low correlation, but a slightly trend showing newest movies have lower mean rating than oldest movies. Despite that, it's clearly visible from the column plot that movies released before 1980 have less ratings than the average of the entire period.

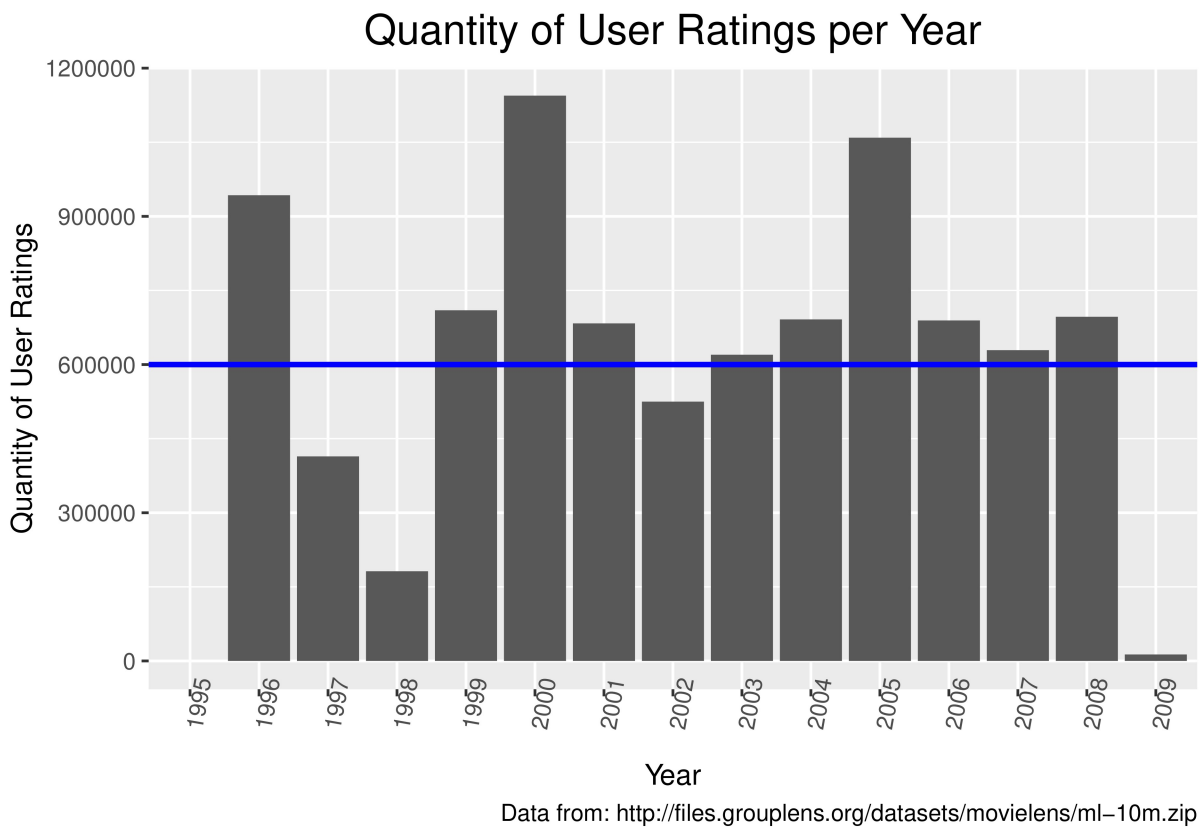
Conclusions:

Oldest movies have less ratings reviews, and for this reason have higher ratings. Despite the fact that the movie release year is few correlated to the movie ratings, it will be added to the models for a better analysis.

- 2.3 Timeastamp Feature Study

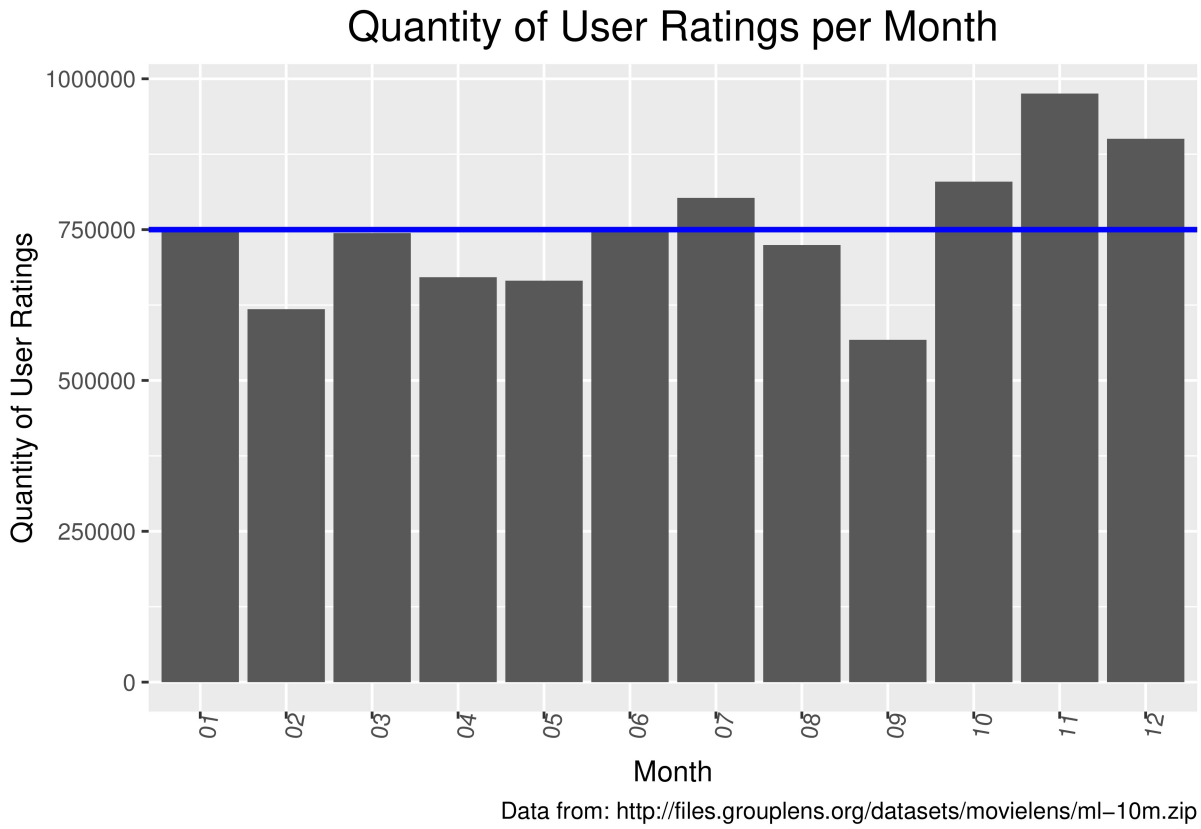
In this section, the timestamp feature will be converted to a date, and we will study if this feature has some correlation to the movies ratings and to other features.

```
# Quantity of ratings per Year
edx %>% group_by(rating_year) %>%
  summarize(count = n()) %>%
  ggplot() +
  geom_col(aes(rating_year, count)) +
  geom_hline(aes(yintercept = mean(count)), size = 1, colour = 'blue', na.rm = TRUE) +
  theme(plot.title = element_text(hjust = 0.5, size = 16),
        axis.text.x = element_text(angle = 80)) +
  labs(title = "Quantity of User Ratings per Year",
       caption = "Data from: http://files.grouplens.org/datasets/movielens/ml-10m.zip") +
  xlab("Year") +
  ylab("Quantity of User Ratings")
```

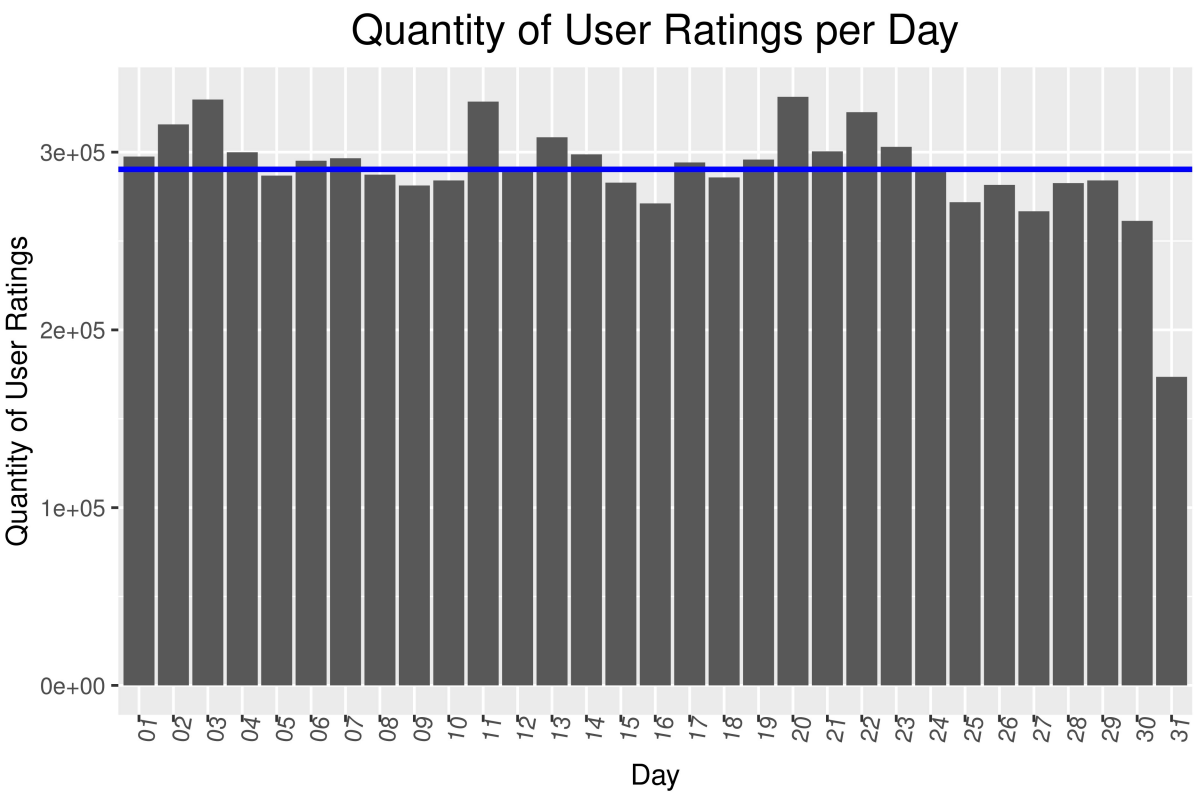



```
# Quantity of ratings per Month
edx %>% group_by(rating_month) %>%
  summarize(count = n()) %>%
```

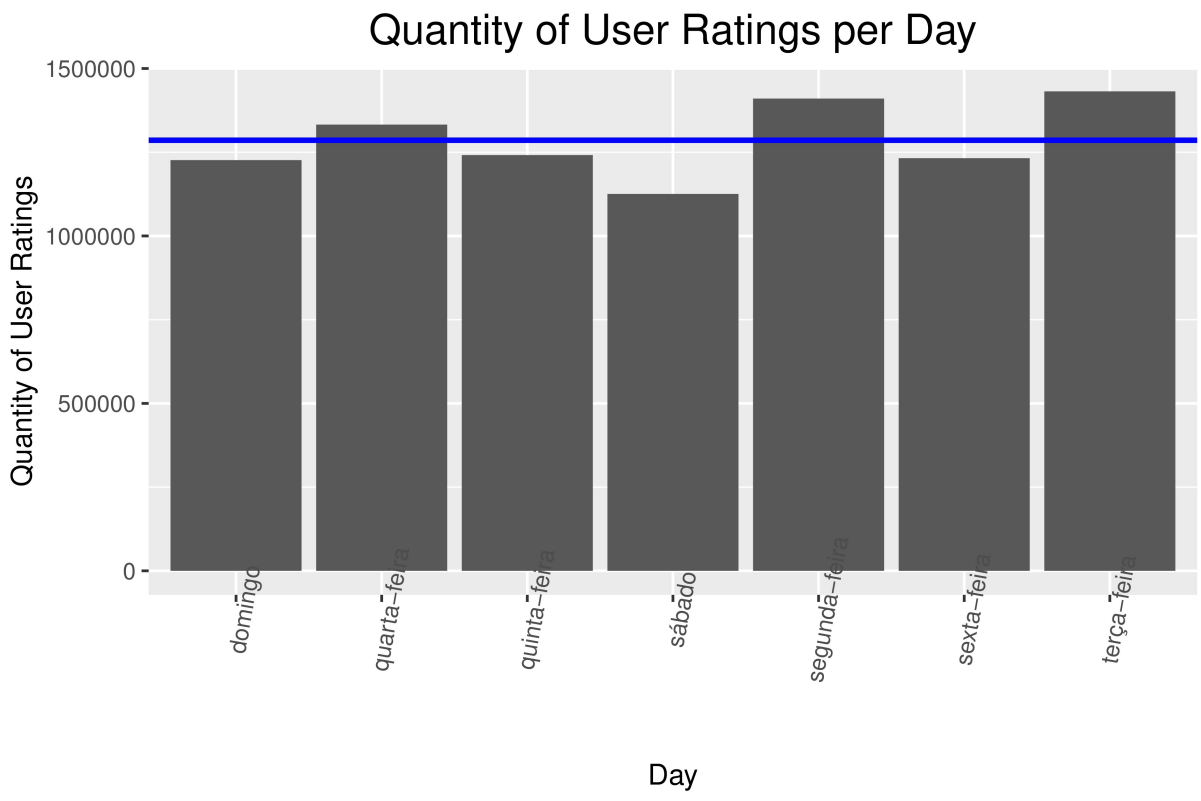
```
  ggplot() +
  geom_col(aes(rating_month, count)) +
  geom_hline(aes(yintercept = mean(count)), size = 1, colour = 'blue', na.rm = TRUE) +
  theme(plot.title = element_text(hjust = 0.5, size = 16),
        axis.text.x = element_text(angle = 80)) +
  labs(title = "Quantity of User Ratings per Month",
       caption = "Data from: http://files.grouplens.org/datasets/movielens/ml-10m.zip") +
  xlab("Month") +
  ylab("Quantity of User Ratings")
```




```
# Quantity of ratings per month-day
edx %>% group_by(rating_mday) %>%
  summarize(count = n()) %>%
  ggplot() +
  geom_col(aes(rating_mday, count)) +
  geom_hline(aes(yintercept = mean(count)), size = 1, colour = 'blue', na.rm = TRUE) +
  theme(plot.title = element_text(hjust = 0.5, size = 16),
        axis.text.x = element_text(angle = 80)) +
  labs(title = "Quantity of User Ratings per Day",
       caption = "Data from: http://files.grouplens.org/datasets/movielens/ml-10m.zip") +
  xlab("Day") +
  ylab("Quantity of User Ratings")
```




```
# Quantity of ratings per week-day
edx %>% group_by(rating_wday) %>%
  summarize(count = n()) %>%
  ggplot() +
  geom_col(aes(rating_wday, count)) +
  geom_hline(aes(yintercept = mean(count)), size = 1, colour = 'blue', na.rm = TRUE) +
  theme(plot.title = element_text(hjust = 0.5, size = 16),
        axis.text.x = element_text(angle = 80)) +
  labs(title = "Quantity of User Ratings per Day",
       caption = "Data from: http://files.grouplens.org/datasets/movielens/ml-10m.zip") +
  xlab("Day") +
  ylab("Quantity of User Ratings")
```



Observations:

From the plot about quantity of user ratings per year, we can observe three predominant peaks on 1996, 2000 and 2005, and four predominant valleys on 1995, 1997, 1998 and 2009, but in general, the distribution floats over the mean.
From the plot about quantity of user ratings per month, we can observe that on July, October, November and December the quantity of ratings overcome the mean line. These are months with important holidays and/or vacations. From the plot about quantity of user ratings per day in a month, we can observe less ratings in the end of the month. It's is probably given that not every month have 31 days.

Data preparation to fit a linear model:

```
# Removing title, timestamp and genres columns
edx <- subset(edx, select = -c(4,5,6,11,12,13,14))
validation <- subset(validation, select = -c(4,5,6,11,12,13,14))

# Data preparing for Linear modelling
edx.numeric <- edx
validation.numeric <- validation

# Changing week day from character to numerical factors
edx.numeric$rating_wday <- as.numeric(as.factor(edx.numeric$rating_wday))
edx.numeric$rating_mday <- as.numeric(edx.numeric$rating_mday)
edx.numeric$rating_month <- as.numeric(edx.numeric$rating_month)
```

```r
edx.numeric$rating_year <- as.numeric(edx.numeric$rating_year)

validation.numeric$rating_wday <- as.numeric(as.factor(validation.numeric$rating_wday))
validation.numeric$rating_mday <- as.numeric(validation.numeric$rating_mday)
validation.numeric$rating_month <- as.numeric(validation.numeric$rating_month)
validation.numeric$rating_year <- as.numeric(validation.numeric$rating_year)

# Changing sub_genres to integer factors
edx.numeric <- edx.numeric %>%
  mutate(genre_1 = as.numeric(as.factor(genre_1)),
         genre_2 = as.numeric(as.factor(genre_2)),
         genre_3 = as.numeric(as.factor(genre_3)),
         genre_4 = as.numeric(as.factor(genre_4)))

validation.numeric <- validation.numeric %>%
  mutate(genre_1 = as.numeric(as.factor(genre_1)),
         genre_2 = as.numeric(as.factor(genre_2)),
         genre_3 = as.numeric(as.factor(genre_3)),
         genre_4 = as.numeric(as.factor(genre_4)))

# Converting NA to -1
edx.numeric$genre_2[is.na(edx.numeric$genre_2)] <- -1
edx.numeric$genre_3[is.na(edx.numeric$genre_3)] <- -1
edx.numeric$genre_4[is.na(edx.numeric$genre_4)] <- -1

validation.numeric$genre_2[is.na(validation.numeric$genre_2)] <- -1
validation.numeric$genre_3[is.na(validation.numeric$genre_3)] <- -1
validation.numeric$genre_4[is.na(validation.numeric$genre_4)] <- -1
```

3. SYSTEM RECOMENDATION LEARNING ALGORITHMS MODELLING:

Linear Model:

The model was trained with all features features

```r
# Linear Regression Fitting
linearRegression.fit <- lm(rating ~ ., data = edx.numeric)
linearRegression.predict <- predict(linearRegression.fit, newdata = validation.numeric)
linearRegression.rmse <- RMSE(linearRegression.predict, validation.numeric$rating)

rmse.results <- tibble(Model = "01. Full Linear Model",
                       RMSE = linearRegression.rmse,
                       Goal = ifelse(linearRegression.rmse < 0.8649, "Under", "Over"))

kable(rmse.results)
```

Model	RMSE	Goal
01. Full Linear Model	1.050587	Over

Observations:

RMSE = 1.0498; Low p-values for all 6 features, which indicates that all of the predictors are associated to the response; R squared = 0.0204, Adjusted R-squared = 0.0204, metric values near to zero indicates that teh regression did not explain much of the variability in the response; Low VIF (Variance Inflation Factor), which indicates low ammount of collinearity between features;

Naive approach:

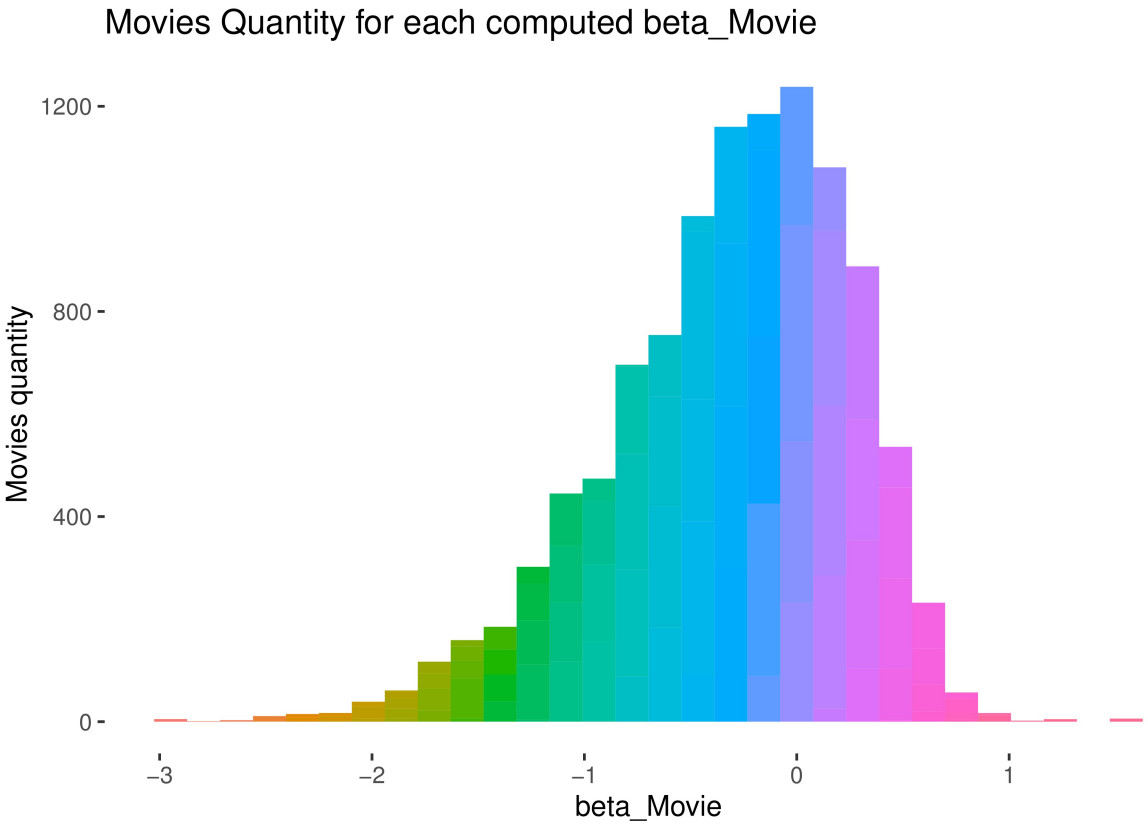
```
mu = mean(edx$rating)

naive.rmse = RMSE(validation$rating, mu)
rmse.results <- rmse.results %>%
  bind_rows(tibble(Model = "02. Mean Rating Approach(Naive)",
                   RMSE = naive.rmse,
                   Goal = ifelse(naive.rmse < 0.8649, "Under", "Over")))
```

Adding to the naive approach the effect of movies in the rating prediction

```
# Adding movie effect to the model
# Rating = mean_rating + beta_movie_effect(movieId)
movie_avg <- edx %>%
  group_by(movieId) %>%
  summarise(b_Movie = mean(rating-mu))

head(movie_avg)
```

```
## # A tibble: 6 x 2
##   movieId b_Movie
##     <dbl>   <dbl>
## 1       1   0.415
## 2       2  -0.307
## 3       3  -0.365
## 4       4  -0.648
## 5       5  -0.444
## 6       6   0.303
```

```
movie_avg %>%
  ggplot(aes(b_Movie, fill = cut(b_Movie, 100))) +
  geom_histogram(bins = 30, show.legend = F) +
  theme_update() +
  labs(x = "beta_Movie", y = "Movies quantity") +
  ggtitle("Movies Quantity for each computed beta_Movie") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "white"))
```




```
mean_movie.predict <- mu + validation %>%
  left_join(movie_avg, by = "movieId") %>%
  pull(b_Movie)

mean_movie.rmse <- RMSE(mean_movie.predict, validation$rating)

rmse.results <- rmse.results %>%
  bind_rows(tibble(Model = "03. Mean Rating + Movie Effect",
                   RMSE = mean_movie.rmse,
                   Goal = ifelse(mean_movie.rmse < 0.8649, "Under", "Over")))

kable(rmse.results)
```

Model	RMSE	Goal
01. Full Linear Model	1.0505874	Over
02. Mean Rating Approach(Naive)	1.0612018	Over
03. Mean Rating + Movie Effect	0.9439087	Over

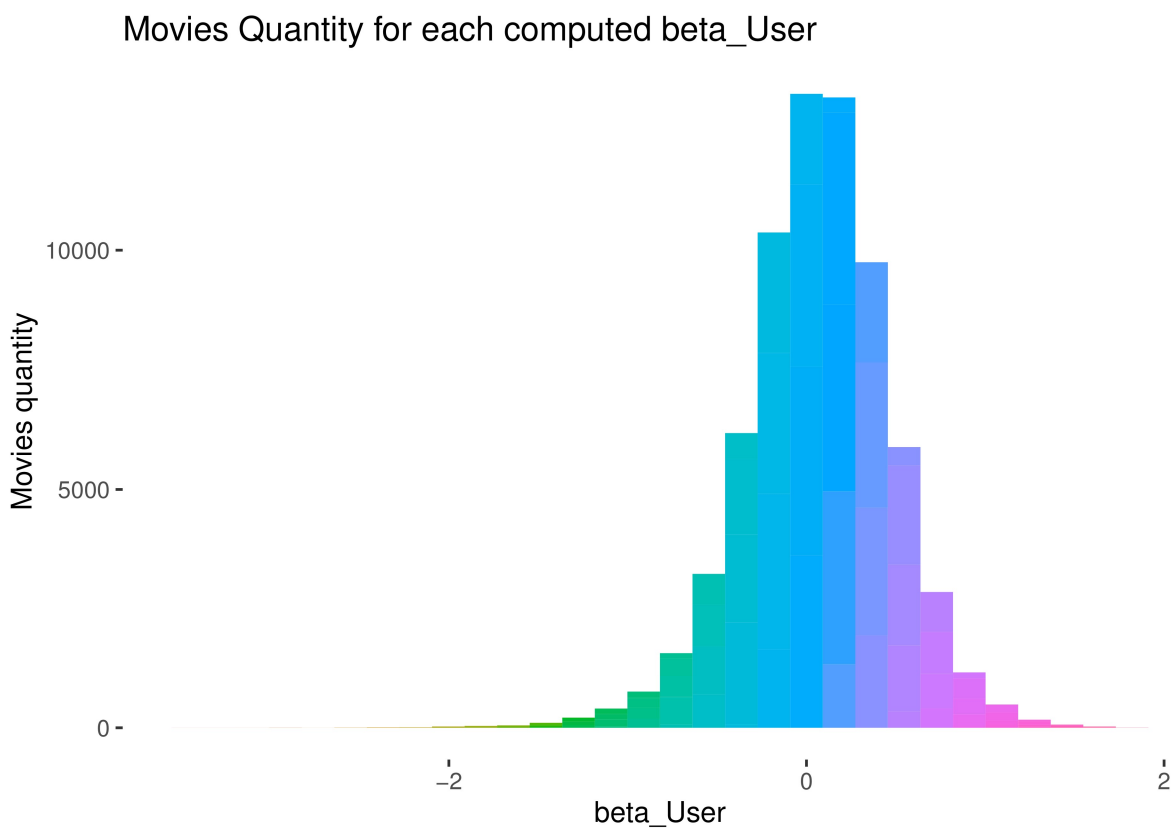
Adding to the previous approach the effect of users in the rating prediction

```
# # Adding user effect to the previous model
# Rating = mean_rating + beta_movie_effect(movieId) + beta_user_effect(userId)
user_avg <- edx %>%
  left_join(movie_avg, by = "movieId") %>%
```

```
  group_by(userId) %>%
  summarise(b_User = mean(rating - b_Movie - mu))

head(user_avg)
```

```
## # A tibble: 6 x 2
##   userId  b_User
##    <int>   <dbl>
## 1      1  1.68
## 2      2 -0.236
## 3      3  0.264
## 4      4  0.652
## 5      5  0.0853
## 6      6  0.346
```

```
user_avg %>%
  ggplot(aes(b_User, fill = cut(b_User, 100))) +
  geom_histogram(bins = 30, show.legend = F) +
  theme_update() +
  labs(x = "beta_User", y = "Movies quantity") +
  ggtitle("Movies Quantity for each computed beta_User") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "white"))
```



```r
mean_movie_user.predict <- validation %>%
  left_join(movie_avg, by = "movieId") %>%
  left_join(user_avg, by = "userId") %>%
  mutate(b_User = mu + b_Movie + b_User) %>%
  pull(b_User)

mean_movie_user.rmse <- RMSE(mean_movie_user.predict, validation$rating)

rmse.results <- rmse.results %>%
  bind_rows(tibble(Model = "04. Mean Rating + Movie Effect + User Effect",
                   RMSE = mean_movie_user.rmse,
                   Goal = ifelse(mean_movie_user.rmse < 0.8649, "Under", "Over")))
kable(rmse.results)
```

Model	RMSE	Goal
01. Full Linear Model	1.0505874	Over
02. Mean Rating Approach(Naive)	1.0612018	Over
03. Mean Rating + Movie Effect	0.9439087	Over
04. Mean Rating + Movie Effect + User Effect	0.8653488	Over

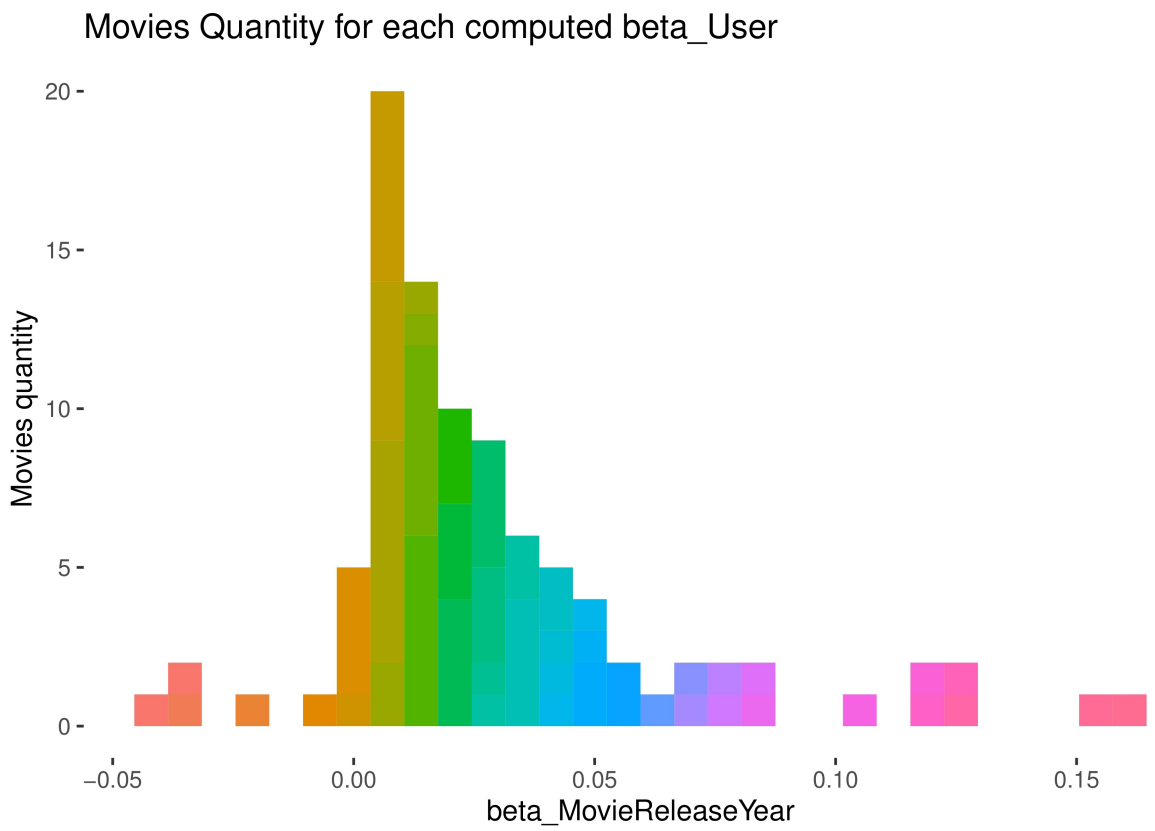
Adding to the previous approach the effect of movie release year in the rating prediction

```r
# Adding release year effect to the previous model
# Rating = mean_rating + beta_movie_effect(movieId) + beta_user_effect(userId) + beta_release_year(movi

MovieReleaseYear_avg <- edx %>%
  left_join(movie_avg, by = "movieId") %>%
  left_join(user_avg, by = "userId") %>%
  group_by(movie_release) %>%
  summarise(b_MRY = mean(rating - b_Movie - b_User - mu))

head(MovieReleaseYear_avg)
```

```
## # A tibble: 6 x 2
##   movie_release  b_MRY
##           <int>  <dbl>
## 1          1915 0.0869
## 2          1916 0.0769
## 3          1917 0.153
## 4          1918 0.0348
## 5          1919 0.164
## 6          1920 0.123
```

```r
MovieReleaseYear_avg %>%
  ggplot(aes(b_MRY, fill = cut(b_MRY, 100))) +
  geom_histogram(bins = 30, show.legend = F) +
  theme_update() +
  labs(x = "beta_MovieReleaseYear", y = "Movies quantity") +
  ggtitle("Movies Quantity for each computed beta_User") +
  theme(panel.grid.major = element_blank(),
```

```
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "white"))
```




```
MovieReleaseYear.predict <- validation %>%
  left_join(movie_avg, by = "movieId") %>%
  left_join(user_avg, by = "userId") %>%
  left_join(MovieReleaseYear_avg, by = "movie_release") %>%
  mutate(b_MRY = mu + b_Movie + b_User + b_MRY) %>%
  pull(b_MRY)

MovieReleaseYear.rmse <- RMSE(MovieReleaseYear.predict, validation$rating)

rmse.results <- rmse.results %>%
  bind_rows(tibble(Model = "05. Mean Rating + Movie Effect + User Effect + Movie Release Year",
                   RMSE = MovieReleaseYear.rmse,
                   Goal = ifelse(MovieReleaseYear.rmse < 0.8649, "Under", "Over")))
kable(rmse.results)
```

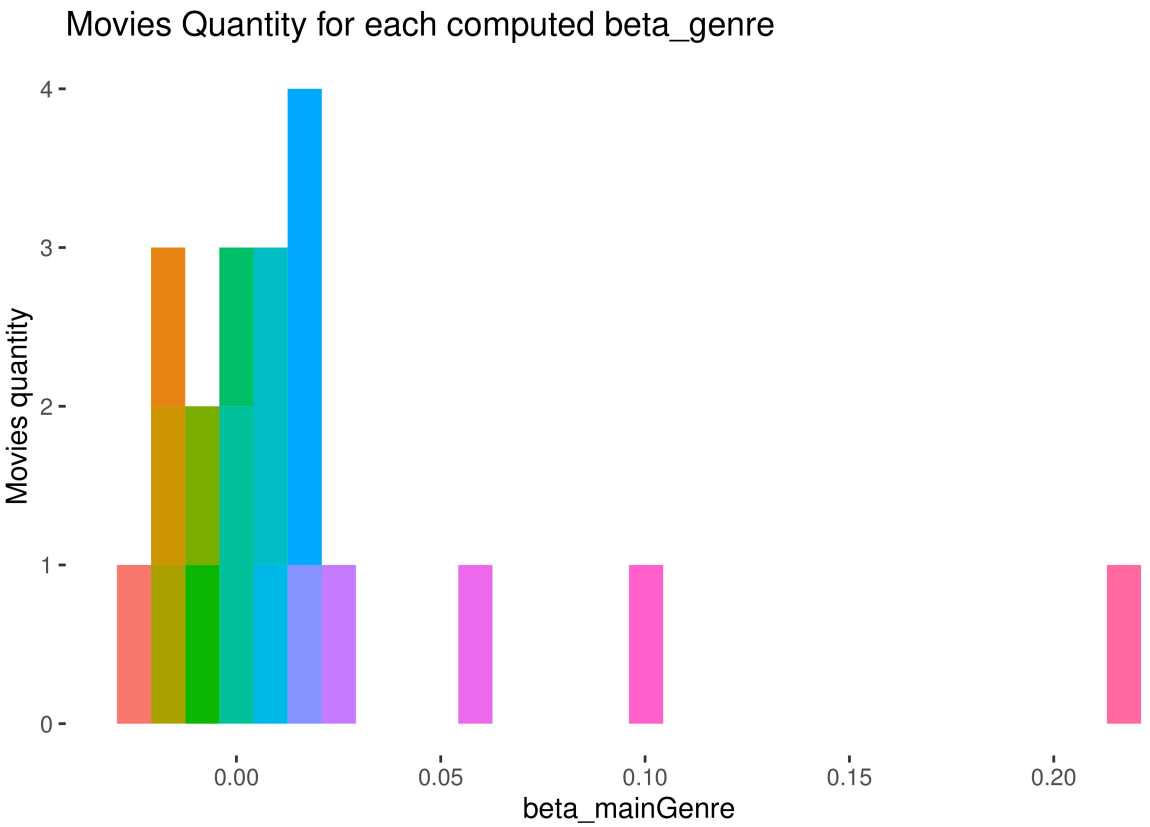
Model	RMSE	Goal
01. Full Linear Model	1.0505874	Over
02. Mean Rating Approach(Naive)	1.0612018	Over
03. Mean Rating + Movie Effect	0.9439087	Over
04. Mean Rating + Movie Effect + User Effect	0.8653488	Over
05. Mean Rating + Movie Effect + User Effect + Movie Release Year	0.8650043	Over

Adding to the previous approach the effect of main movie genre (genre_1) in the rating prediction

```
# # Adding main genre effect to the previous model
# Rating = mean_rating + beta_movie_effect(movieId) + beta_user_effect(userId) + beta_release_year(movi

mainGenre_avg <- edx %>%
  left_join(movie_avg, by = "movieId") %>%
  left_join(user_avg, by = "userId") %>%
  left_join(MovieReleaseYear_avg, by = "movie_release") %>%
  group_by(genre_1) %>%
  summarise(b_genre = mean(rating - b_Movie - b_User - b_MRY - mu))

head(mainGenre_avg)
```

```
## # A tibble: 6 x 2
##   genre_1             b_genre
##   <chr>                 <dbl>
## 1 (no genres listed)  0.218
## 2 Action             -0.0121
## 3 Adventure          -0.0129
## 4 Animation          -0.0159
## 5 Children           -0.0242
## 6 Comedy              0.00342
```

```
mainGenre_avg %>%
  ggplot(aes(b_genre, fill = cut(b_genre, 100))) +
  geom_histogram(bins = 30, show.legend = F) +
  theme_update() +
  labs(x = "beta_mainGenre", y = "Movies quantity") +
  ggtitle("Movies Quantity for each computed beta_genre") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "white"))
```




```
mainGenre.predict <- validation %>%
  left_join(movie_avg, by = "movieId") %>%
  left_join(user_avg, by = "userId") %>%
  left_join(MovieReleaseYear_avg, by = "movie_release") %>%
  left_join(mainGenre_avg, by = "genre_1") %>%
  mutate(b_genre = mu + b_Movie + b_User + b_MRY + b_genre) %>%
  pull(b_genre)

mainGenre.rmse <- RMSE(mainGenre.predict, validation$rating)

rmse.results <- rmse.results %>%
  bind_rows(tibble(Model = "06. Mean Rating + Movie Effect + User Effect + Movie Release Year + Genre E
                   RMSE = mainGenre.rmse,
                   Goal = ifelse(mainGenre.rmse < 0.8649, "Under", "Over")))
kable(rmse.results)
```

Model	RMSE	Goal
01. Full Linear Model	1.0505874	Over
02. Mean Rating Approach(Naive)	1.0612018	Over
03. Mean Rating + Movie Effect	0.9439087	Over
04. Mean Rating + Movie Effect + User Effect	0.8653488	Over
05. Mean Rating + Movie Effect + User Effect + Movie Release Year	0.8650043	Over
06. Mean Rating + Movie Effect + User Effect + Movie Release Year + Genre Effecf	0.8649057	Over