HarvardX Data Science Final Project - Recommendation Systems

Jakub Garncarek

2023-03-17

1. Introduction

This project was inspired by Netflix challenge. In October 2006, Netflix offered a challenge on improving their movie recommendation system by 10% to win a prize money of a million dollars. Recommendation systems use ratings that the users give an item on buying and/or using them to make specific recommendations.Companies like Amazon collect massive data sets of user ratings on the products sold to them. The data sets are subsequently used to predict a high rated items for a given user and are then recommended to the user. Similarly, Netflix use movie ratings provided by users to predict a high rated movie for a given user and then recommended it to the user. Usually the movie ratings are on a 1-5 scale, where 5 represents an excellent movie and 1 suggests it to be a poor one. In this project I explored MovieLens data set which contains over 10 millions movie ratings. Data set was created by researchers at the University of Minnesota. The goal was to develop machine-learning model to predict movie ratings in test set achieving RMSE less than 0.86490. RMSE can be interpret similar to standard deviation.

2. Data preparation and loading libraries

```
library(tidyr)
library(tidyverse)
library(dplyr)
library(caret)
library(ggplot2)
library(lubridate)
library(knitr)
```

```
dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)
```

```
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))

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


set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
# set.seed(1) # if using R 3.5 or earlier
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]


final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

removed <- anti_join(temp, final_holdout_test)
edx <- rbind(edx, removed)

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

Extract the year of production of movie from title and create new column - year_of_prod

```
edx <- edx %>% mutate(year_of_prod = str_extract(title, pattern= ".\\d{4}.$")) %>%
  mutate(year_of_prod = str_extract(year_of_prod, pattern = "\\d{4}"))
```

Replace timestamp to rate dates and round them to month

```
edx <- edx %>% mutate(rate_date = as_datetime(timestamp)) %>% select(-timestamp)
edx$rate_date <- round_date(edx$rate_date, unit="month")
```

Split edx data to avoid overfitting. In orther words i don't want testing models final test set because this may causing that best model will be matched specifically to test data and produce incorrect results

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
edx_test <- edx[test_index,]
edx_train <- edx[-test_index,]
```

Make sure movieId and userId in test data are also in train data

```
edx_test <- edx_test %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")
```

3. Data analysis

Head of train set

```
kable(tibble(head(edx_train)))
```

userId	movieId	rating	title	genres	year_of_prod	date_date
1	122	5	Boomerang (1992)	Comedy|Romance	1992	1996-08-01
1	185	5	Net, The (1995)	Action|Crime|Thriller	1995	1996-08-01
1	292	5	Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1995	1996-08-01
1	316	5	Stargate (1994)	Action|Adventure|Sci-Fi	1994	1996-08-01
1	329	5	Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1994	1996-08-01
1	355	5	Flintstones, The (1994)	Children|Comedy|Fantasy	1994	1996-08-01

Number of rows of train set

```
nrow(edx_train)
```

```
## [1] 8100048
```

Number of columns of train set

```
ncol(edx_train)
```

```
## [1] 7
```

3.1 Users analysis

Number of unique users in train set

```
edx_train %>% summarize(number_of_users = length(unique(userId)))
```

```
##   number_of_users
## 1           69878
```

Create object user_rates which contains number of rates and mean rate for each user

```
user_rates <- edx_train %>% group_by(userId) %>%
  summarize(num_of_rates = n(), mean_rate = mean(rating))
```

Plot number of users versus number of rates

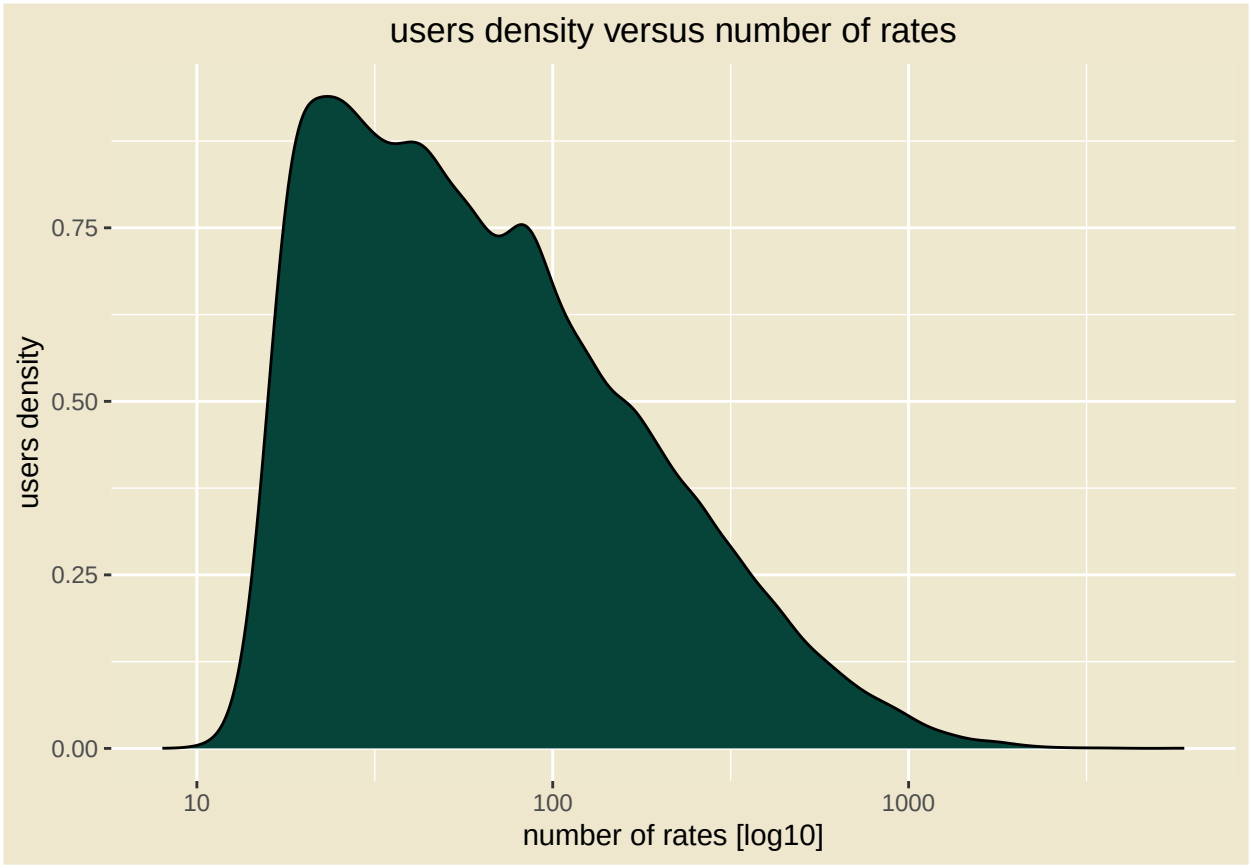


Chart above isnt't look normal even with x axis in log scale. Values are shifted to left site. It means that most of users give relatively small number of rates. In that case to calculation number of rates that standard user give median works much better than average.

```
median(user_rates$num_of_rates)
```

```
## [1] 56
```

```
mean(user_rates$num_of_rates)
```

```
## [1] 115.917
```

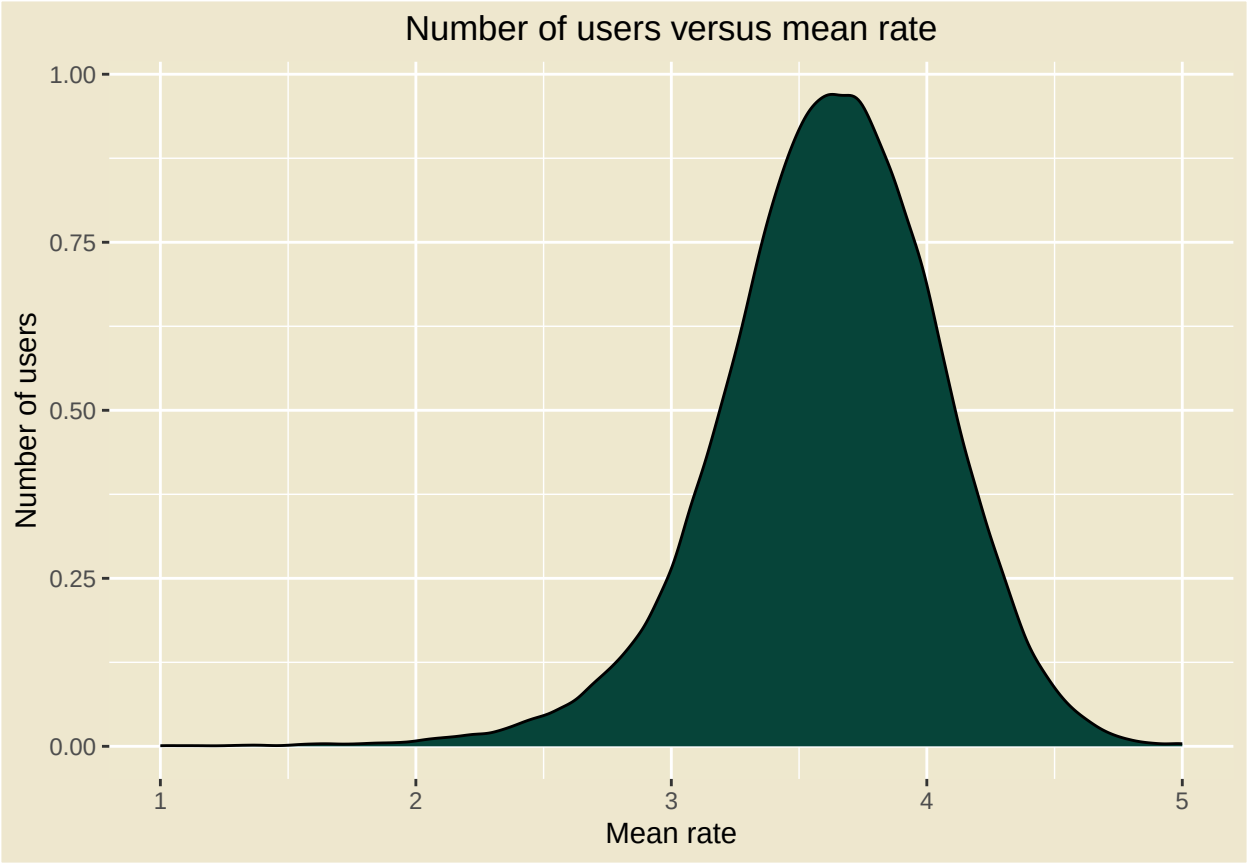
To confirm that in this case median works better calculate proporcion users who posted more than mean rates

```
nrow(filter(user_rates, num_of_rates > mean(user_rates$num_of_rates)))/nrow(user_rates)
```

```
## [1] 0.2734051
```

Plot count of users versus average rate

```
## Warning: Removed 6 rows containing non-finite values (`stat_density()`).
```



Density of number of users looks quite normal so median and average values should be almost equal

```
median(user_rates$mean_rate)
```

```
## [1] 3.633501
```

```
mean(user_rates$mean_rate)
```

```
## [1] 3.613548
```

3.2 Movie analysis

Number of unique movies in train set

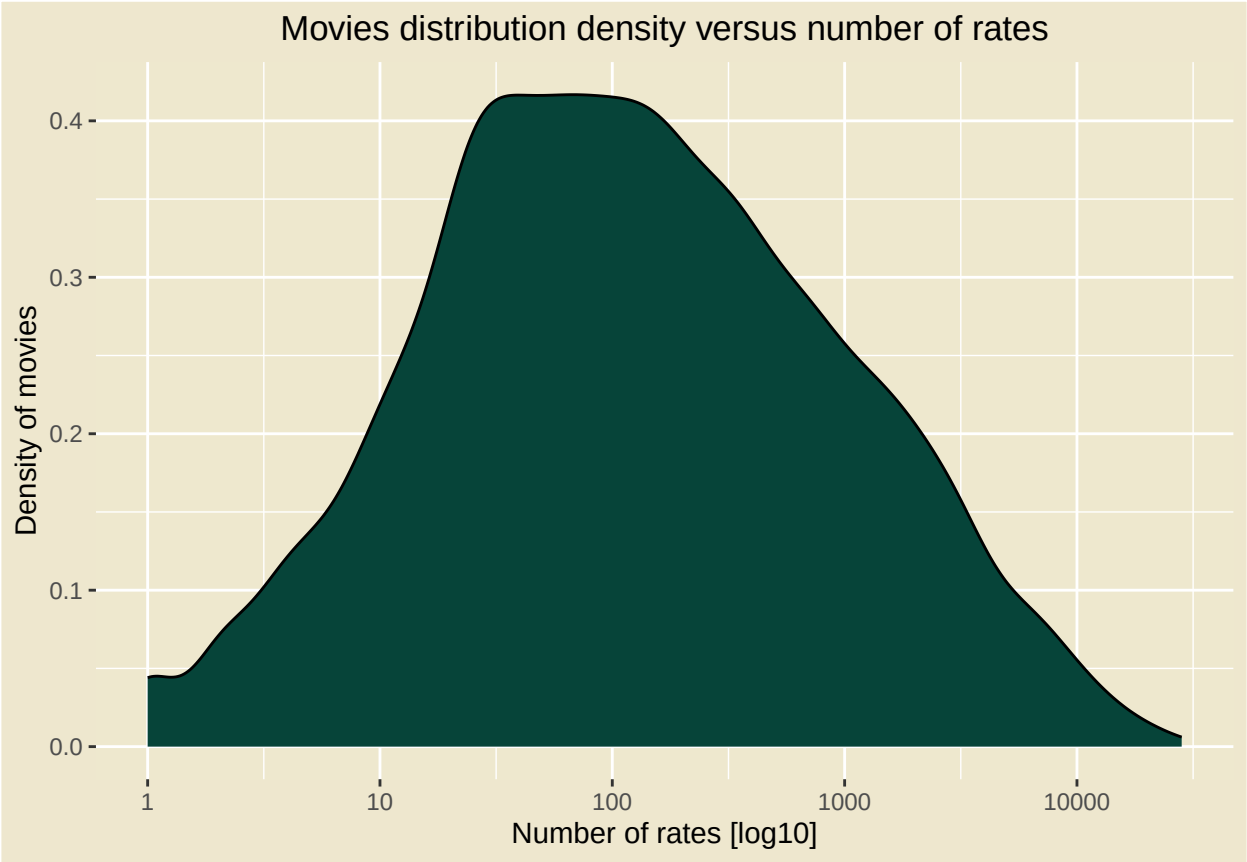
```
edx_train %>% summarize(number_of_movies = length(unique(movieId)))
```

```
##   number_of_movies
## 1            10656
```

Create object movies_rates which contain mean rate and number of rates for each movie

```
movies_rates <- edx_train %>% group_by(movieId) %>%
  summarize(mean_rate = mean(rating), num_of_rates=n())
```

Plot movies distribution density versus number of rates



Movies distribution density looks quite normal with x axis in log10 scale. It means that most of movies have relative small number of rate and median calculate number of rates to standard movie better than average

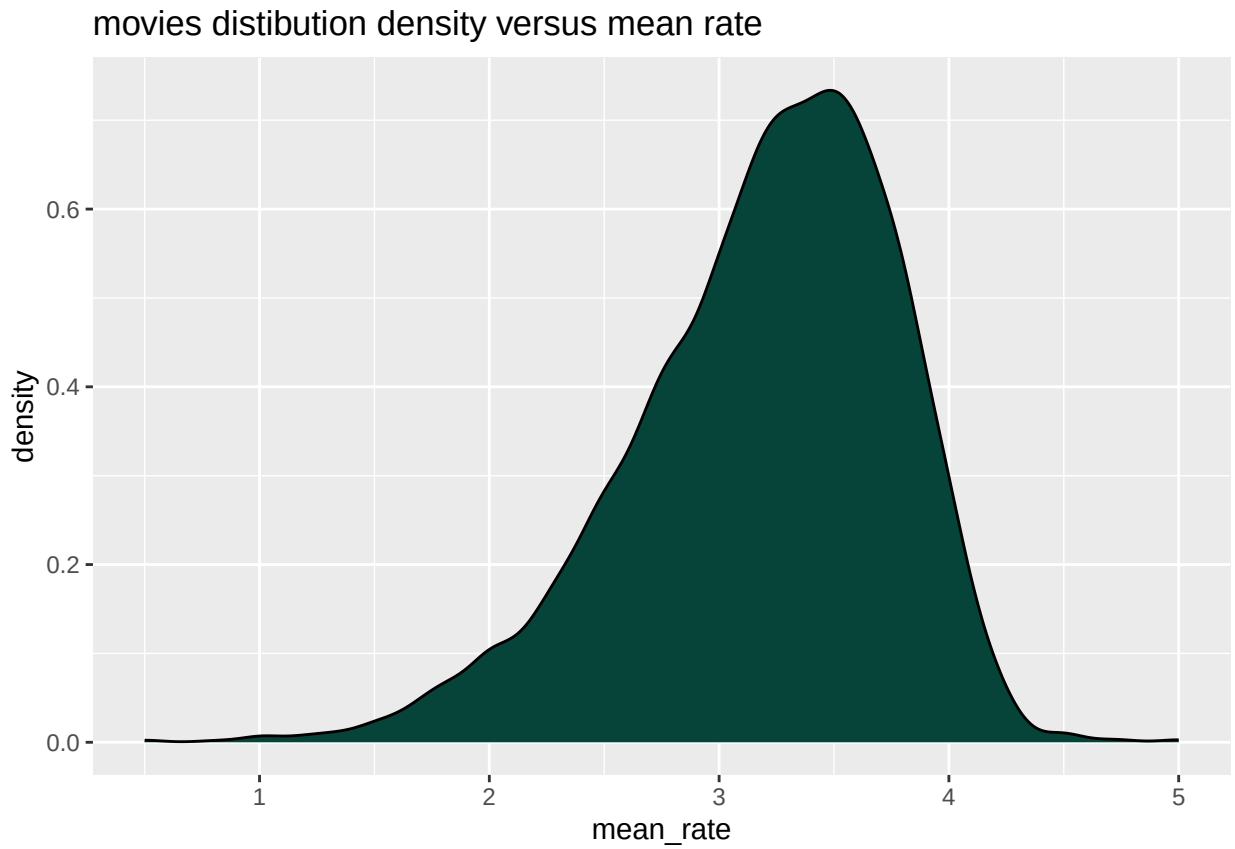
```
mean(movies_rates$num_of_rates)
```

```
## [1] 760.1396
```

```
median(movies_rates$num_of_rates)
```

```
## [1] 110
```

Plot movies distribution density versus mean rate




```
## NULL
```

Plot movie mean rate versus number of rates

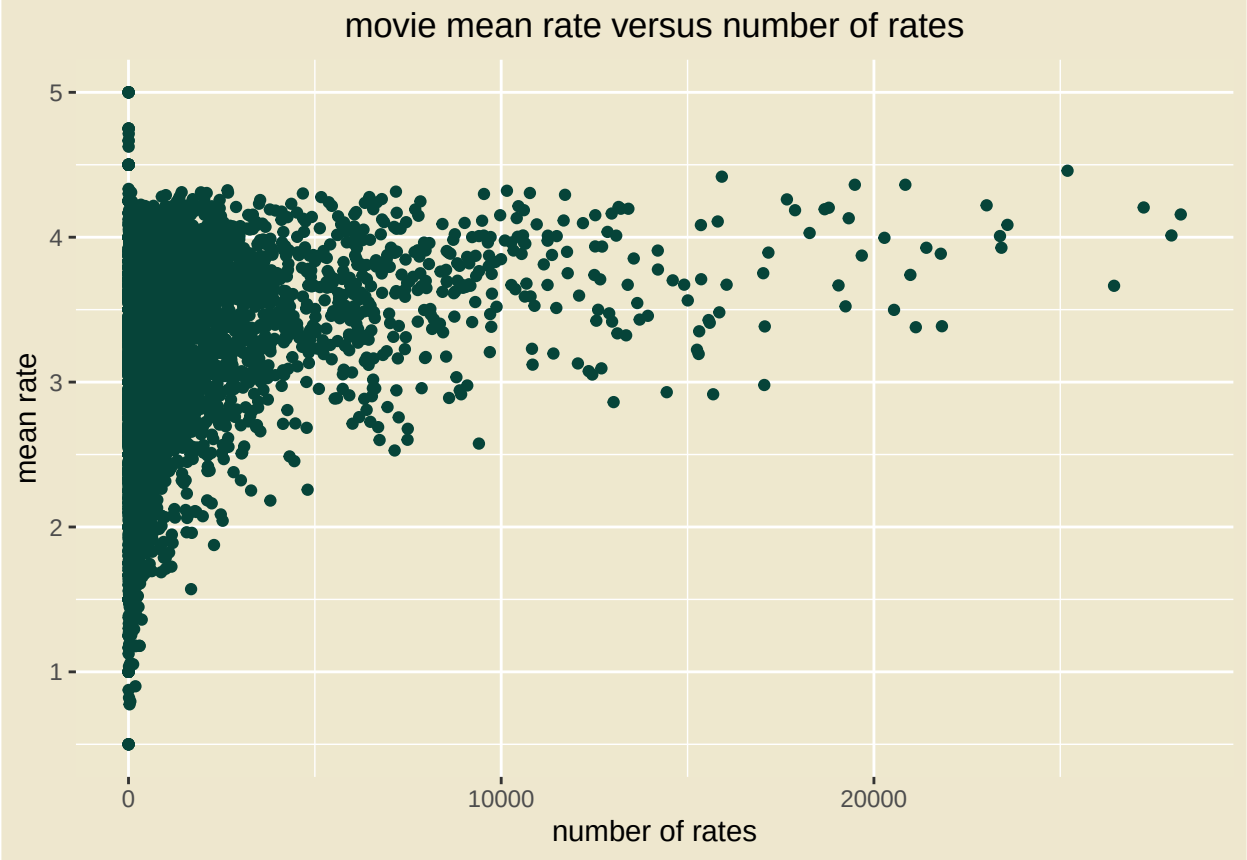


Chart shows that with increase number of rates to movie value standard deviation of thier mean rates decrease. This is expected phenomenon, because most of people watch well-rated movies, so they are rated more often

Best rated movies

```
movies_rates <- edx_train %>% group_by(movieId) %>%
  filter(n()>500) %>% summarize(mean_rate = mean(rating)) %>%
  arrange(desc(mean_rate))

top_movie <- top_n(movies_rates,10)

top_movie %>%
  left_join(edx_train, by="movieId") %>%
  summarize(unique(title), unique(movieId), unique(mean_rate)) %>% kable()
```

unique(title)	unique(movieId)	unique(mean_rate)
Shawshank Redemption, The (1994)	318	4.458980
Godfather, The (1972)	858	4.417808
Schindler's List (1993)	527	4.362112
Usual Suspects, The (1995)	50	4.362069
Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	922	4.323928
Casablanca (1942)	912	4.320662
Rear Window (1954)	904	4.314660

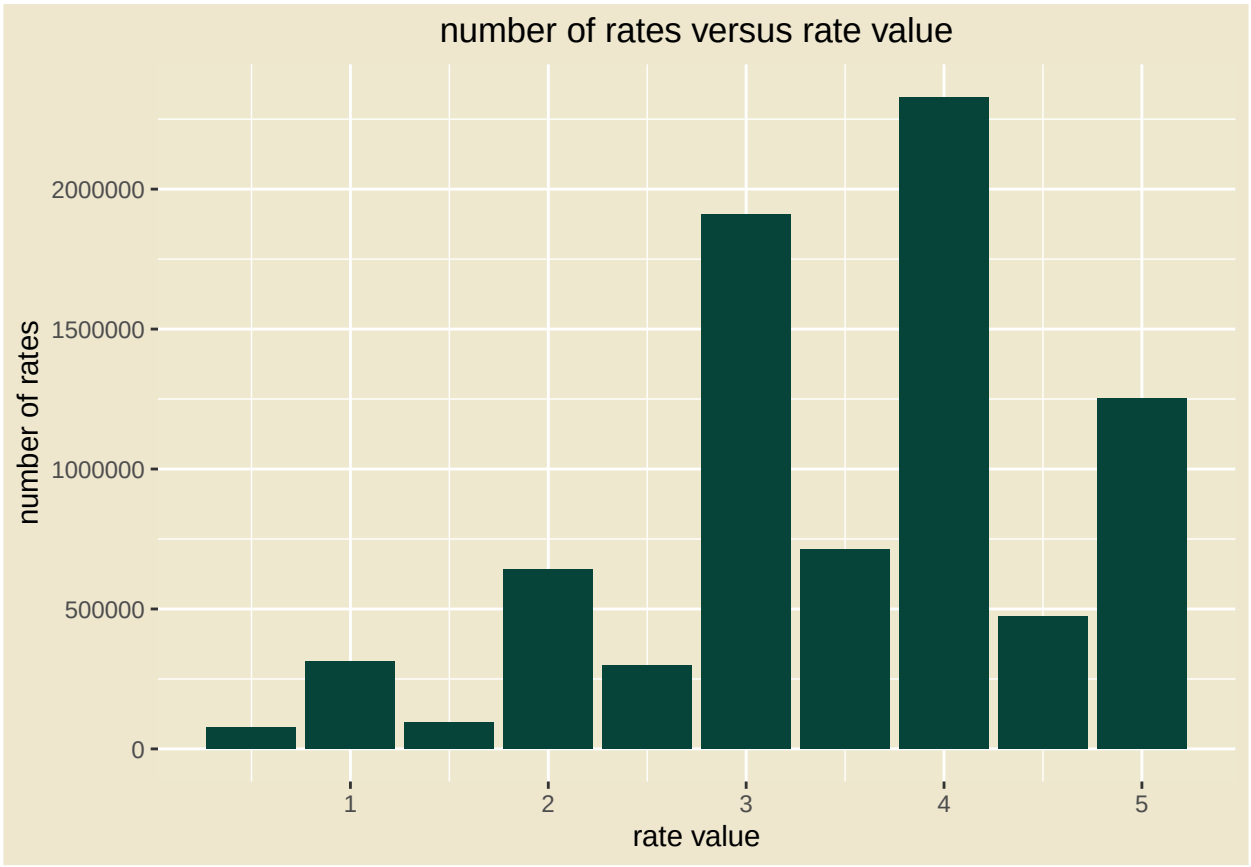
unique(title)	unique(movieId)	unique(mean_rate)
Paths of Glory (1957)	1178	4.310659
Double Indemnity (1944)	3435	4.309671
Third Man, The (1949)	1212	4.307750

3.3 Ranking analysis

Create object rates which contain number of rates for each rate value

```
rates <- edx_train %>% group_by(rating) %>% summarize(n = n())
```

Plot number of rates versus rate value



This chart shows that people are more likely to give integer rates than non-integer. It may be caused integer looks less complicated and people subconsciously choose them more often. This phenomenon can be reduce by set range of rate value to 1-10 without decimal values.

3.4 Production year anasysis

Create object years_rates which contain number of rate and mean rate which received movies created in each year and number of this movies

```
years_rates <- edx_train %>% group_by(year_of_prod) %>%
  summarize(num_of_rates =n(), mean_rating = mean(rating), num_of_movies = length(unique(movieId)))
```

Plot number of rates versus year of movie production

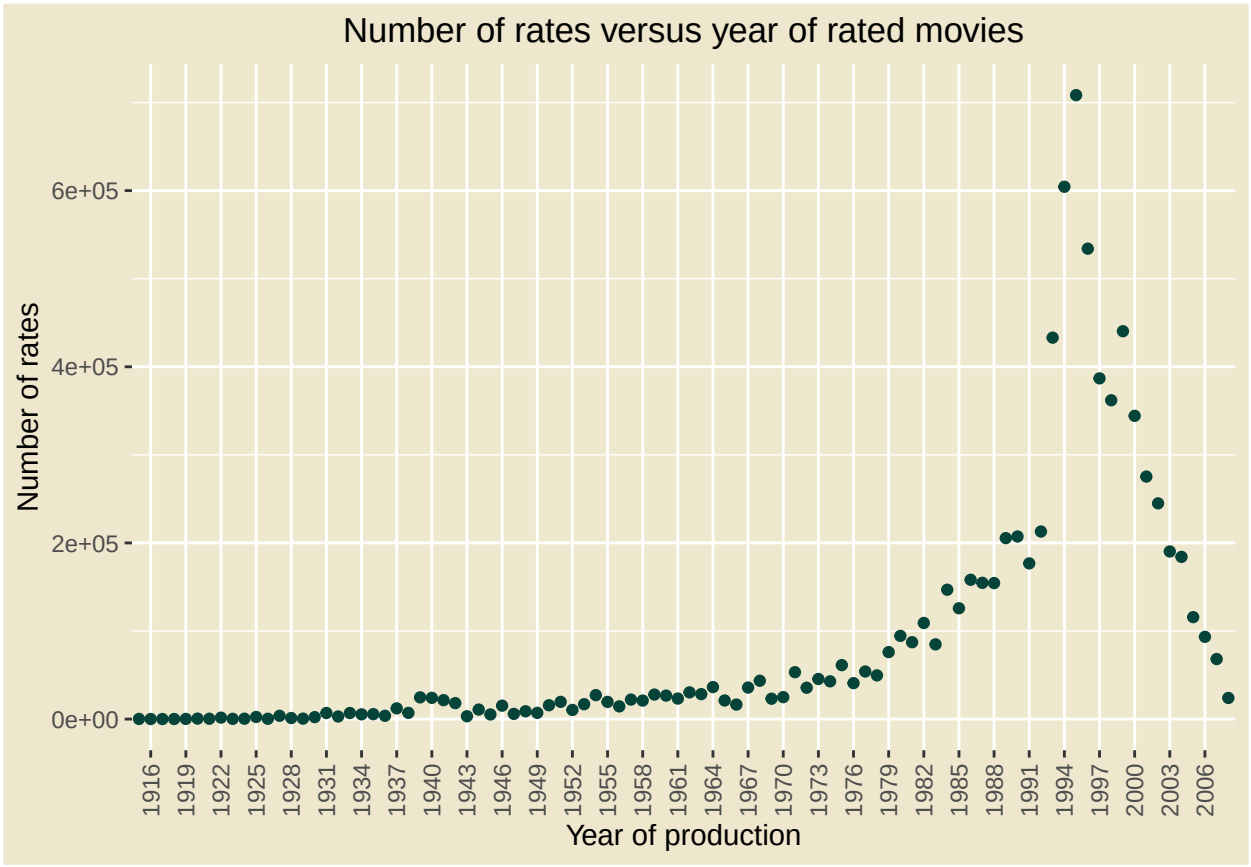
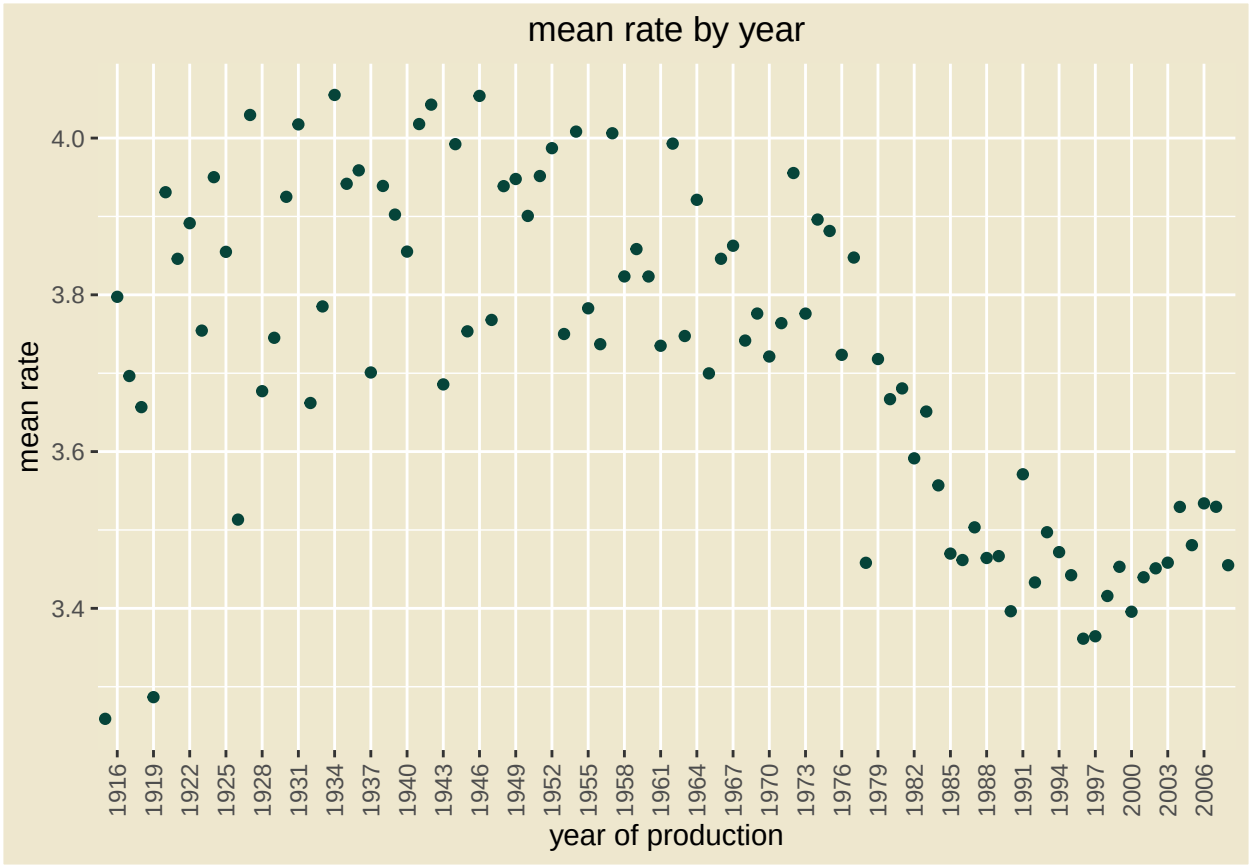


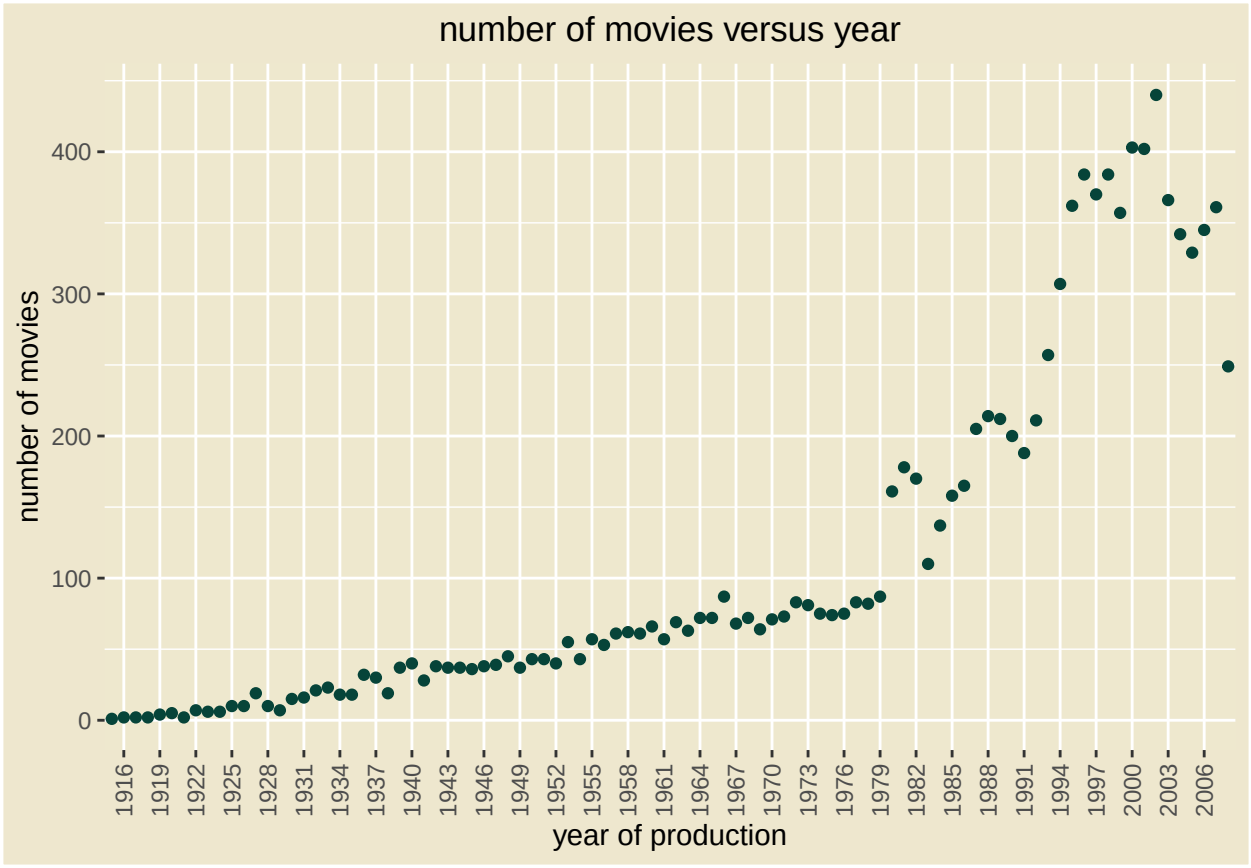
Chart shows huge pic of mean number of rates for movie created about 90's. In next years trend is downward. This is probably because people have had less time to watch movies made in recent years.

Plot mean annual rating versus year of movie production

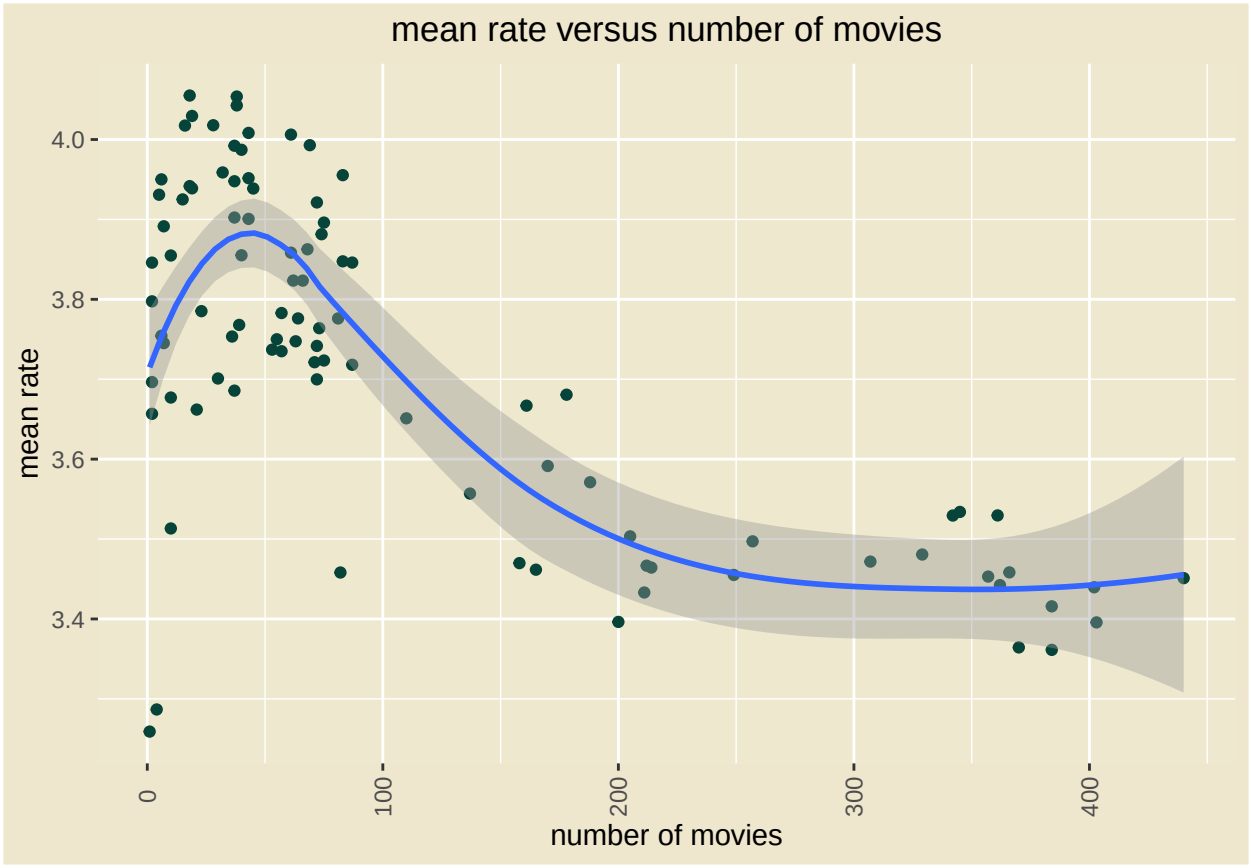


Can be noticed that mean annual rating decreases after 70's. A closer look at the data is needed to explain this trend. To do this plot number of movies that was created in each year by year and mean annual rating by number of movies created in each year.

Plot number of movies versus year

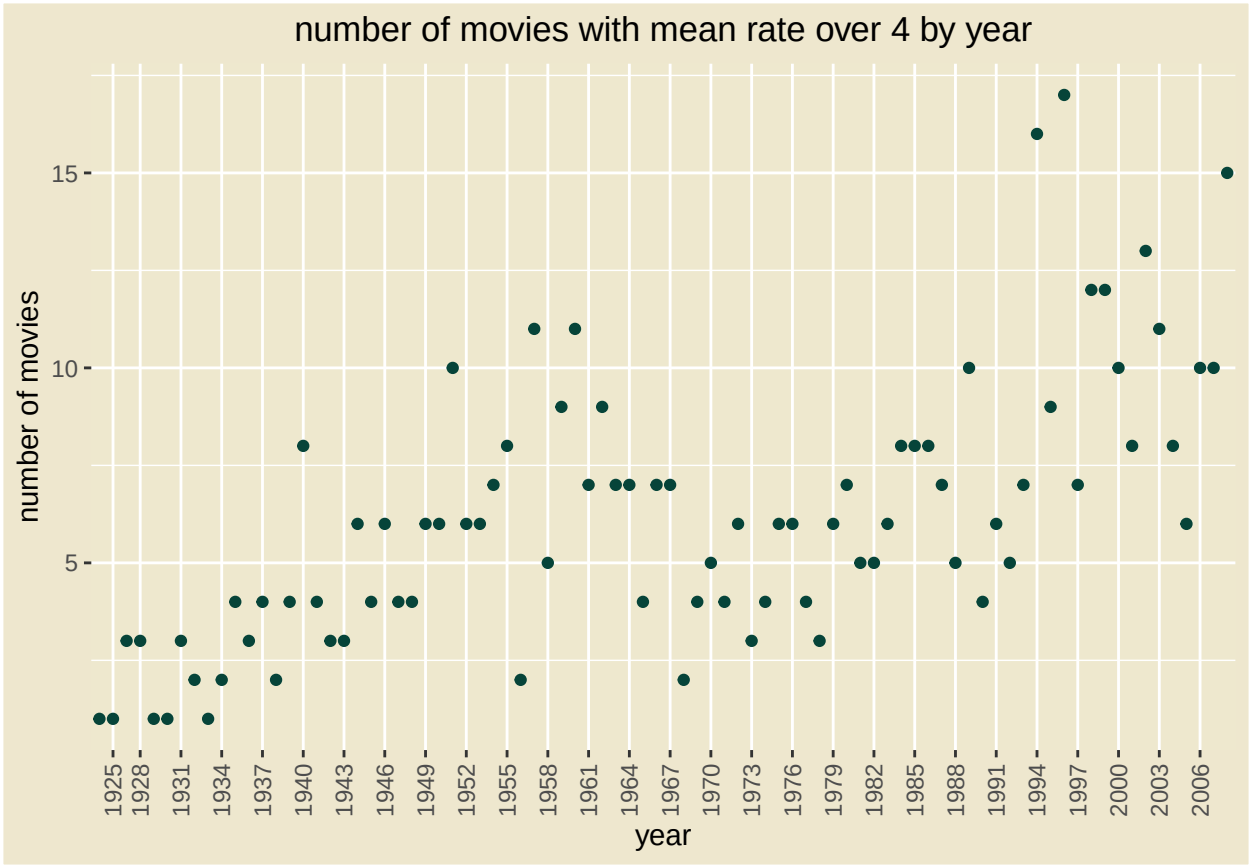


Plot mean annual rate versus number of movies



First chart shows that after 70's annual number of movie productions has increased significantly. From secound chart can be noticed that when annual number of movie productions increased mean anual rating decreased. To be sure what it's exactly means, extract number of best movies from each year(those which mean rating is over 4) and plot them by year.

```
movies_rates <- edx_train %>% group_by(movieId) %>%
  summarize(mean_rate = mean(rating), num_of_rates=n())

best_year_movies <- edx_train %>% left_join(movies_rates, by="movieId") %>%
  filter(mean_rate >=4) %>% group_by(year_of_prod) %>%
  summarize(best_movies = length(unique(movieId)))
```



The number of high rated movie even increased after 70's. It's mean that annual number of movie production doesn't affect to number of really good movies, but inceased number of medium and bad movies which lowers annual mean rate

3.5 Genres analysis

Can be noticed that part of movies have more than one caregory

```
kable(head(edx_train$genres))
```

x
Comedy|Romance
Action|Crime|Thriller
Action|Drama|Sci-Fi|Thriller
Action|Adventure|Sci-Fi
Action|Adventure|Drama|Sci-Fi
Children|Comedy|Fantasy

Number of unique genres combos in train set

```
edx_train %>% summarize(genres_combo = length(unique(genres)))
```

```
##   genres_combo
## 1          797
```

Create object genres by select genres and rating column and separate multiple genres in rows

```
genres <- edx_train %>% select(genres,rating) %>% separate_rows(genres, sep = "\\|")
```

Number of single genres in train set

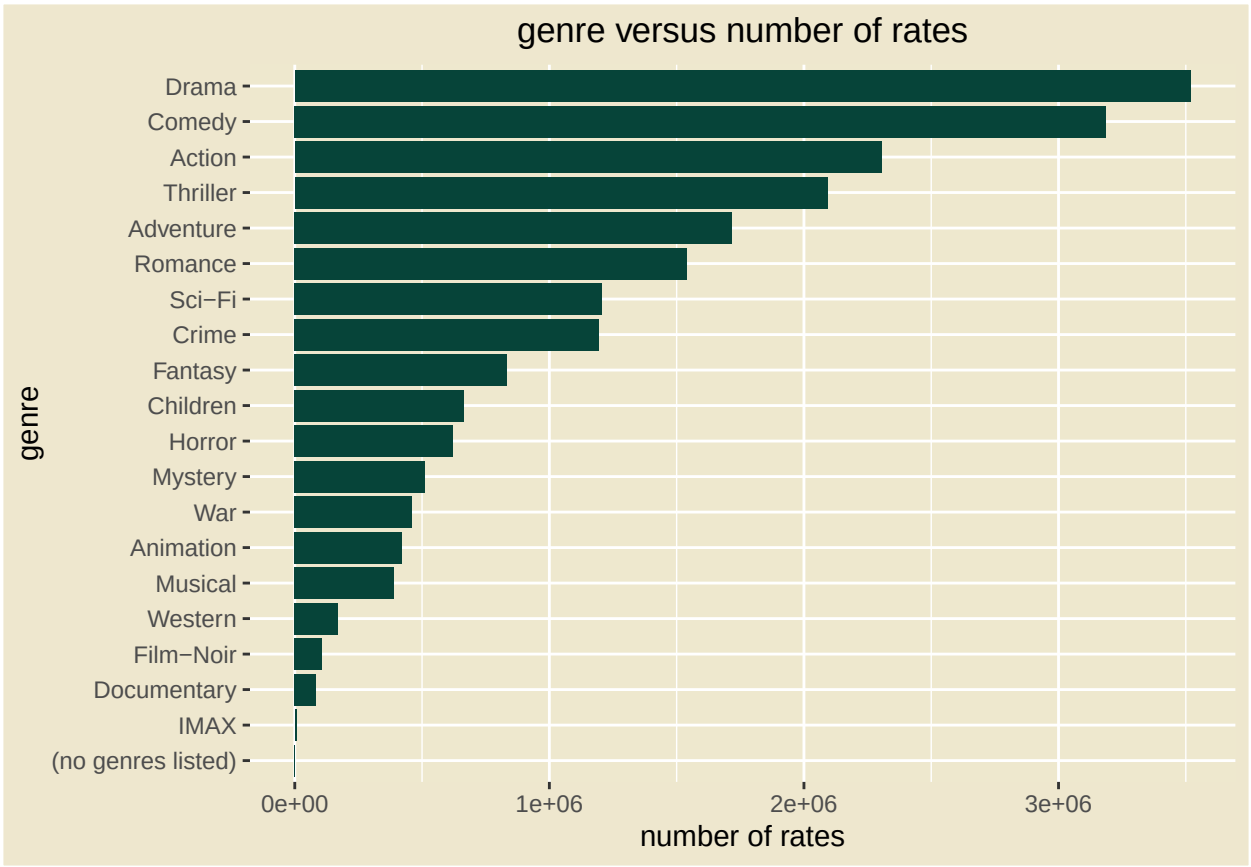
```
genres %>% summarize(number_of_genres = length(unique(genres)))
```

```
## # A tibble: 1 x 1
##   number_of_genres
##              <int>
## 1               20
```

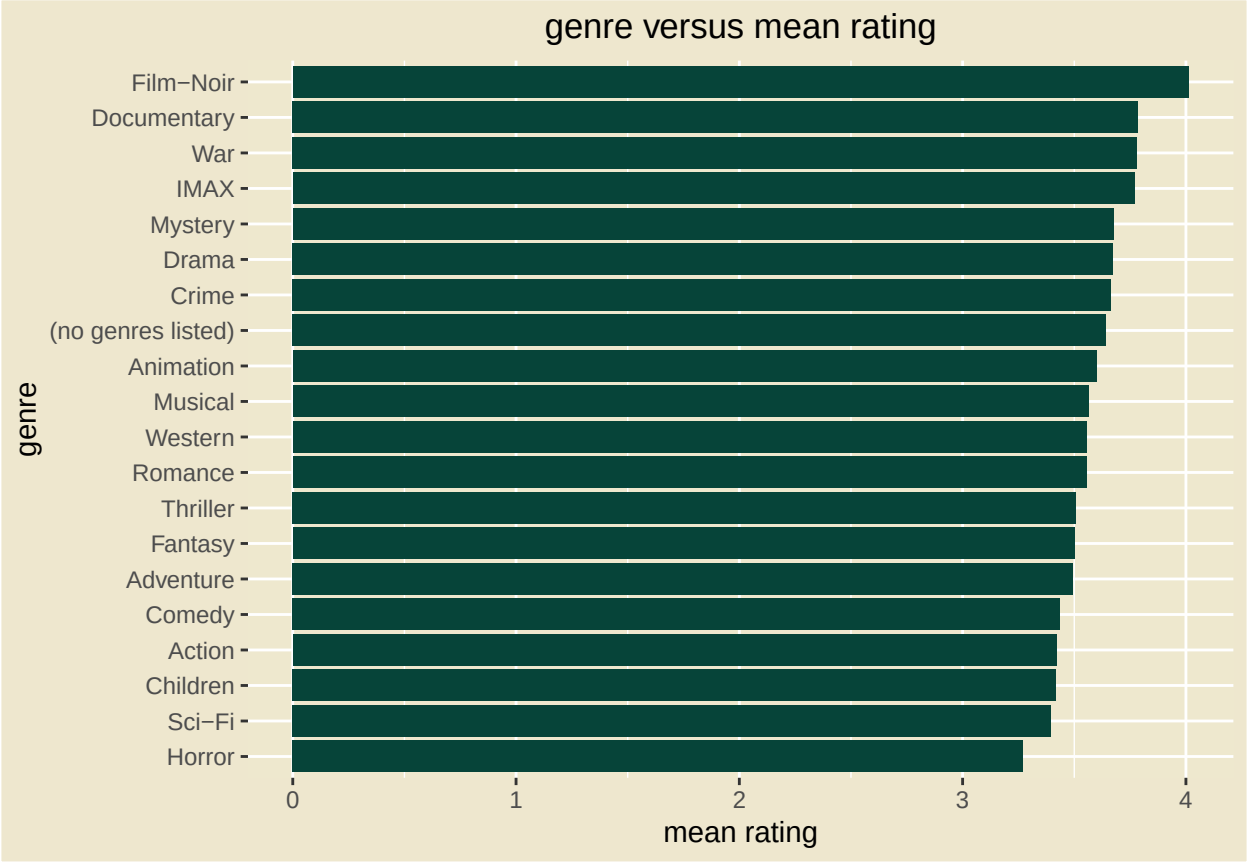
Create genres rating object which contain number of rates and mean rating for single genre

```
genres_rating <- genres %>% group_by(genres) %>%
  summarize(num_of_rates = n(), mean_rating = mean(rating))
```

Plot genre versus number of rates



Plot genre versus mean rating

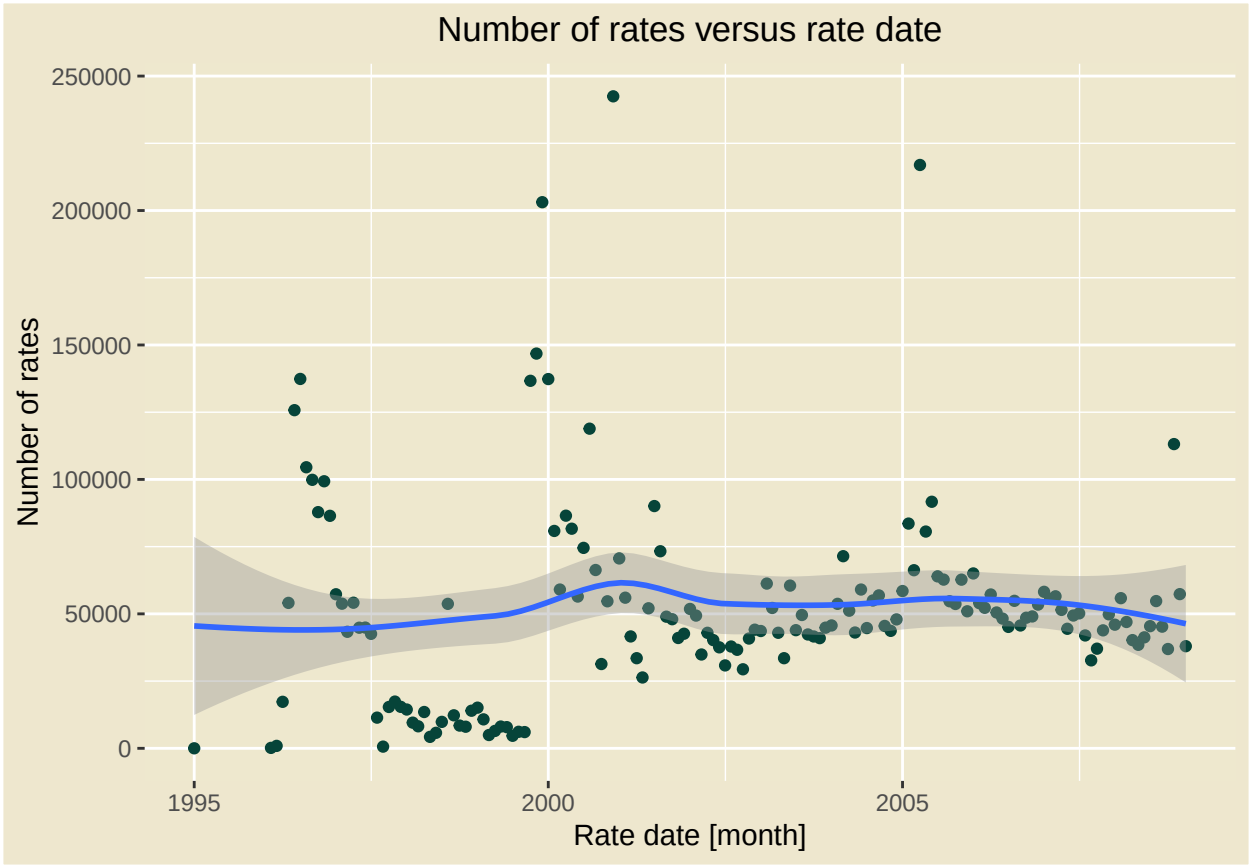


3.6 Rate date analysis

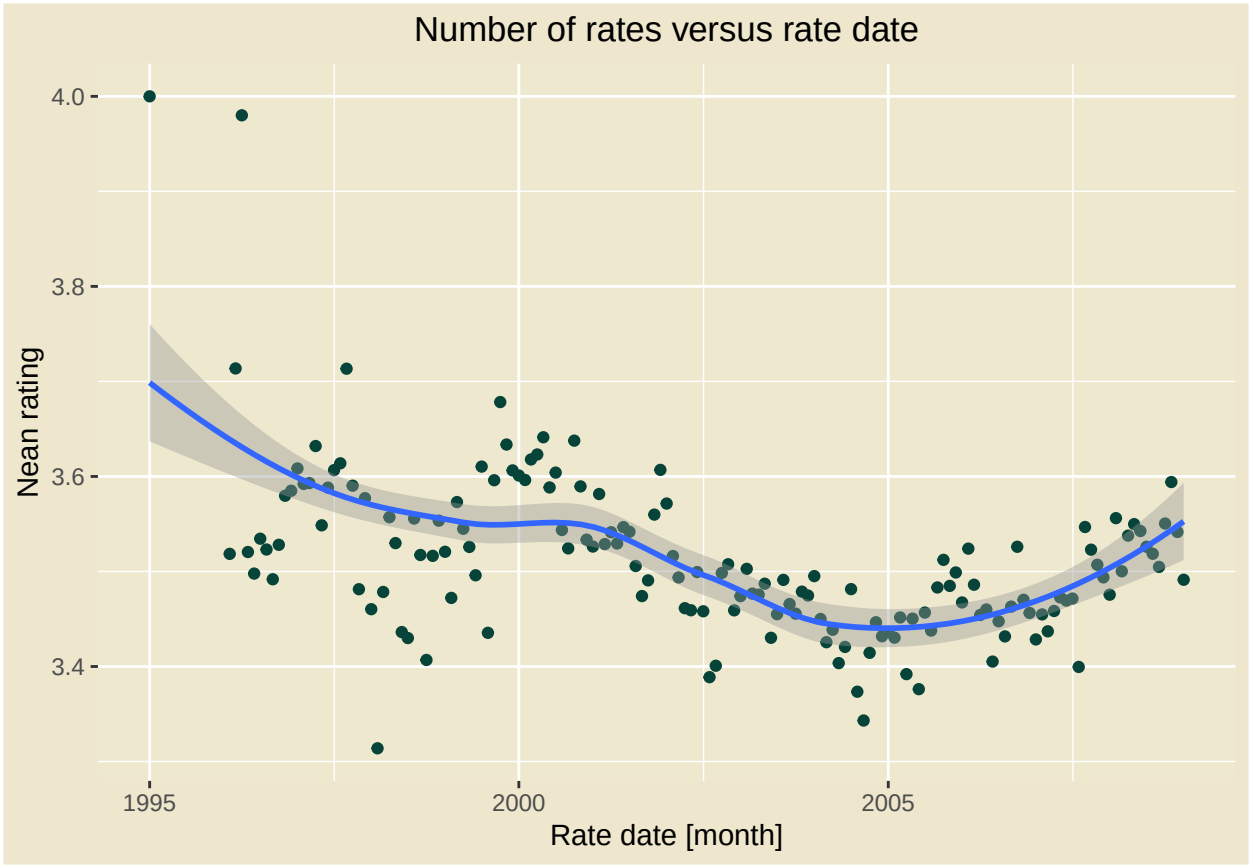
Create object dates_rating which contain number of rates and mean rate that users post in each month

```
dates_rating <- edx_train %>%
  group_by(rate_date) %>% summarize(n = n(), mean_rating = mean(rating))
```

Plot number of rates versus rate date



Plot mean rate versus rate date



4. Creting models

Loss function:To compare different models or to see how well we're doing compared to some baseline, the typical error loss, the residual mean squared error (RMSE) is used on the test set. We can interpret RMSE similar to standard deviation.

```
RMSE <- function(y_hat, y){
  sqrt(mean((y_hat-y)^2))
}
```

4.1. Mean base model

Building the first model- the simplest possible recommendation system: This model assumes the same rating for all movies and users with all the differences explained by random variation. The model is as follows:

```
edx_average <- mean(edx_train$rating)
mean_m1 <- RMSE(edx_average, edx_test$rating)
```

results:

```
kable(tibble(Model_Type = c("mean base model"),
                         RMSE = c(mean_m1))%>%
                           mutate(RMSE = sprintf("%0.6f", RMSE)))
```

Model_Type	RMSE
mean base model	1.060333

4.2 Modeling movie effect

Improve model by adding movie effect

```
movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - edx_average))

predicted_ratings <- edx_average + edx_test %>%
  left_join(movie_avgs, by="movieId") %>% pull(b_i)

movie_m2 <- RMSE(predicted_ratings, edx_test$rating)
```

Results:

```
kable(tibble(Model_Type = c("mean base model", "Movie effect model"),
                         RMSE = c(mean_m1, movie_m2)) %>%
                           mutate(RMSE = sprintf("%0.6f", RMSE)))
```

Model_Type	RMSE
mean base model	1.060333
Movie effect model	0.943626

4.3 Modeling movie + user effect

Improve model by adding user effect

```
user_avgs <- edx_train %>%
  left_join(movie_avgs, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - edx_average - b_i))

predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by = "movieId") %>%
  left_join(user_avgs, by = "userId") %>%
  mutate(pred = edx_average + b_i + b_u) %>% pull(pred)

movie_user_m3 <- RMSE(predicted_ratings, edx_test$rating)
```

Results:

```
kable(tibble(Model_Type = c("mean base model", "Movie effect model",
                            "Movie + user effect model"),
             RMSE = c(mean_m1, movie_m2, movie_user_m3))%>%
                        mutate(RMSE = sprintf("%0.6f", RMSE)))
```

Model_Type	RMSE
mean base model	1.060333
Movie effect model	0.943626
Movie + user effect model	0.865629

4.4 Modeling movie + user + year effect

Improve model by adding year effect

```
year_avgs <- edx_train %>%
  left_join(user_avgs, by="userId") %>%
  left_join(movie_avgs, by="movieId") %>%
  group_by(year_of_prod) %>%
  summarize(b_a = mean(rating - edx_average - b_u - b_i))

predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by ="userId") %>%
  left_join(year_avgs, by="year_of_prod") %>%
  mutate(pred = edx_average + b_a + b_u + b_i)%>%
  pull(pred)

movie_user_year_m4 <- RMSE(predicted_ratings, edx_test$rating)
```

Results:

```
kable(tibble(Model_Type = c("mean base model", "Movie effect model",
                            "Movie + user effect model",
                            "Movie + user + year effect model"),
             RMSE = c(mean_m1, movie_m2, movie_user_m3,
                              movie_user_year_m4))%>%
                        mutate(RMSE = sprintf("%0.6f", RMSE)))
```

Model_Type	RMSE
mean base model	1.060333
Movie effect model	0.943626
Movie + user effect model	0.865629
Movie + user + year effect model	0.865309

4.5 Movie + user + year + rate date effect

Improve model by adding rate date effect

```
rate_date_avgs <- edx_train %>%
  left_join(movie_avgs, by = "movieId") %>%
  left_join(user_avgs, by="userId") %>%
  left_join(year_avgs, by="year_of_prod") %>%
  group_by(rate_date) %>%
  summarize(d = smooth(mean(rating - edx_average - b_u -b_i - b_a)))

predicted_ratings <- edx_test %>%

  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by ="userId") %>%
  left_join(year_avgs, by="year_of_prod") %>%
  left_join(rate_date_avgs, by="rate_date") %>%
  mutate(pred = edx_average + b_u +b_i + d +b_a) %>%
  pull(pred)

movie_user_year_ratedate_m5 <- RMSE(predicted_ratings, edx_test$rating)
```

Results:

```
kable(tibble(Model_Type = c("mean base model", "Movie effect model",
                            "Movie + user effect model",
                            "Movie + user + year effect model",
                            "movie + user + year + rate date effect"),
             RMSE = c(mean_m1, movie_m2, movie_user_m3, movie_user_year_m4,
                      movie_user_year_ratedate_m5)) %>%
                          mutate(RMSE = sprintf("%0.6f", RMSE)))
```

Model_Type	RMSE
mean base model	1.060333
Movie effect model	0.943626
Movie + user effect model	0.865629
Movie + user + year effect model	0.865309
movie + user + year + rate date effect	0.865224

4.6 Movie + user + year + rate date + genre effect

Improve model by adding genre effect

```
genres_avgs <- edx_train %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by="userId") %>%
  left_join(year_avgs, by="year_of_prod") %>%
  left_join(rate_date_avgs, by="rate_date") %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>% summarize(b_g = mean(rating- edx_average - b_i - b_u -b_a -d))
```

```
genres_combo <- edx_train %>% select(genres) %>%
  group_by(genres) %>% summarize() %>%
  mutate(id = seq.int(n()))
```

```
genres_combo_avgs <- genres_combo %>%
  separate_rows(genres,sep = "\\|") %>%
  left_join(genres_avgs, by="genres") %>% group_by(id) %>%
  summarize(b_g = mean(b_g)) %>%
  left_join(genres_combo, by="id") %>% select(-id)
```

```
predicted_ratings <-  edx_test %>%
  left_join(movie_avgs, by="movieId") %>%
  left_join(user_avgs, by="userId") %>%
  left_join(year_avgs, by="year_of_prod") %>%
  left_join(rate_date_avgs, by="rate_date") %>%
  left_join(genres_combo_avgs, by="genres") %>%
  mutate(pred = edx_average + b_i + b_u + b_g +b_a +d) %>% pull(pred)

movie_user_year_ratedate_genres_m6 <- RMSE(predicted_ratings, edx_test$rating)
```

Results:

```
kable(tibble(Model_Type = c("mean base model", "Movie effect model",
                            "Movie + user effect model",
                            "Movie + user + year effect model",
                            "movie + user + year + rate date effect",
                            "movie + user + year + rate date + genre effect"),
             RMSE = c(mean_m1, movie_m2, movie_user_m3, movie_user_year_m4,
                      movie_user_year_ratedate_m5,
                      movie_user_year_ratedate_genres_m6))%>%
                          mutate(RMSE = sprintf("%0.6f", RMSE)))
```

Model_Type	RMSE
mean base model	1.060333
Movie effect model	0.943626
Movie + user effect model	0.865629
Movie + user + year effect model	0.865309
movie + user + year + rate date effect	0.865224
movie + user + year + rate date + genre effect	0.865131

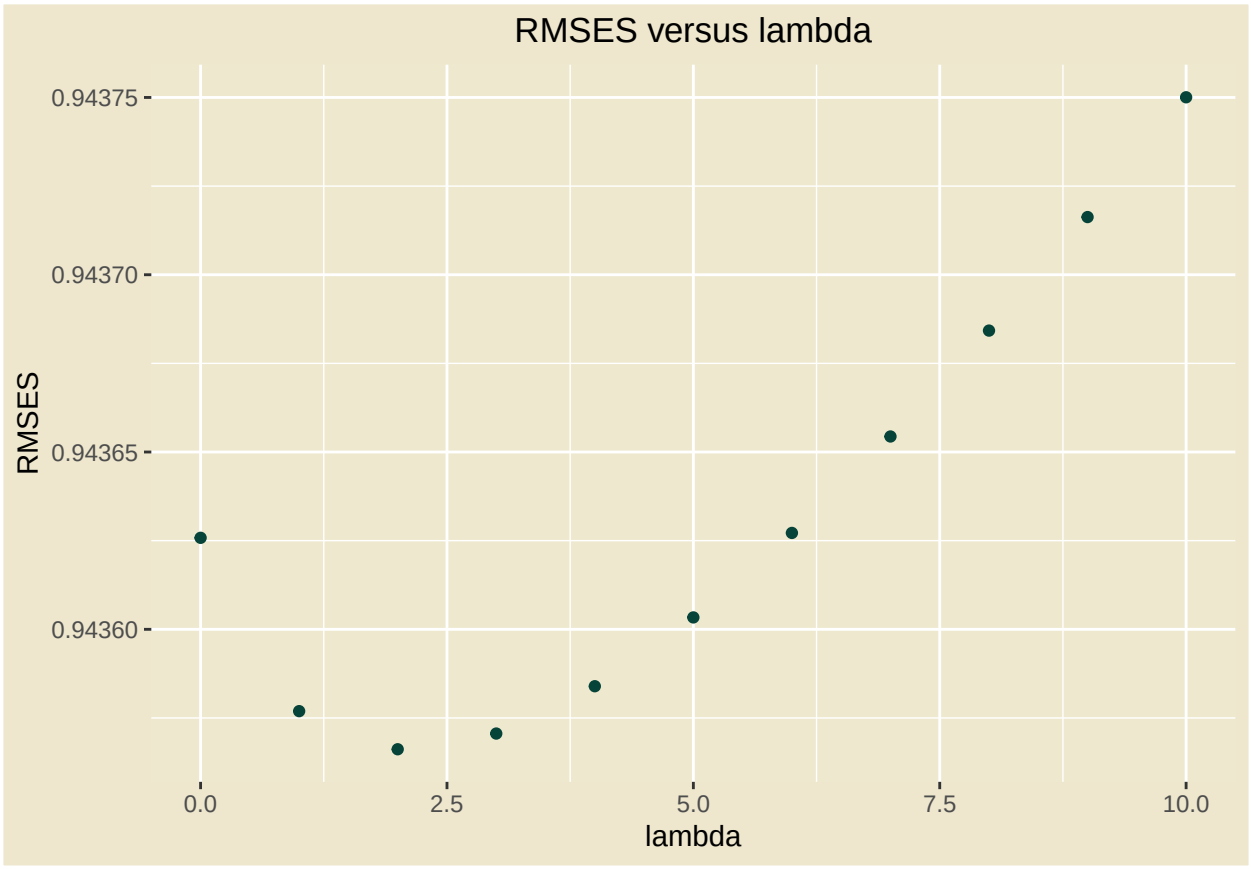
4.7 Movie effect with regularization

```
lambda <- seq(0,10,1)

edx_average <- mean(edx_train$rating)


RMSES <- sapply(lambda, function(l){
```

```
  movie_reg <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - edx_average)/(n()+l))

  predicted_ratings <- edx_average + edx_test %>%
    left_join(movie_reg, by="movieId") %>%
    pull(b_i)

  return(RMSE(predicted_ratings, edx_test$rating))
})

movie_reg_m2 <- min(RMSES)

lambda[which.min(RMSES)]
```

```
## [1] 2
```

Plot RMSES versus lambda



Results:

```
kable(tibble(Model_Type = c("mean base model", "Movie effect model",
                            "Movie + user effect model",
                            "Movie + user + year effect model",
```

```
                          "movie + user + year + rate date effect",
                          "movie + user + year + rate date + genre effects"),
            RMSE = c(mean_m1, movie_m2, movie_user_m3, movie_user_year_m4,
                     movie_user_year_ratedate_m5, movie_user_year_ratedate_genres_m6),
            RMSE_with_reg = c(NA,movie_reg_m2,NA,NA,NA,NA)) %>%
                        mutate(RMSE = sprintf("%0.6f", RMSE),
                               RMSE_with_reg = sprintf("%0.6f", RMSE_with_reg)))
```

Model_Type	RMSE	RMSE_with_reg
mean base model	1.060333	NA
Movie effect model	0.943626	0.943566
Movie + user effect model	0.865629	NA
Movie + user + year effect model	0.865309	NA
movie + user + year + rate date effect	0.865224	NA
movie + user + year + rate date + genre effects	0.865131	NA

Redefine movie effect

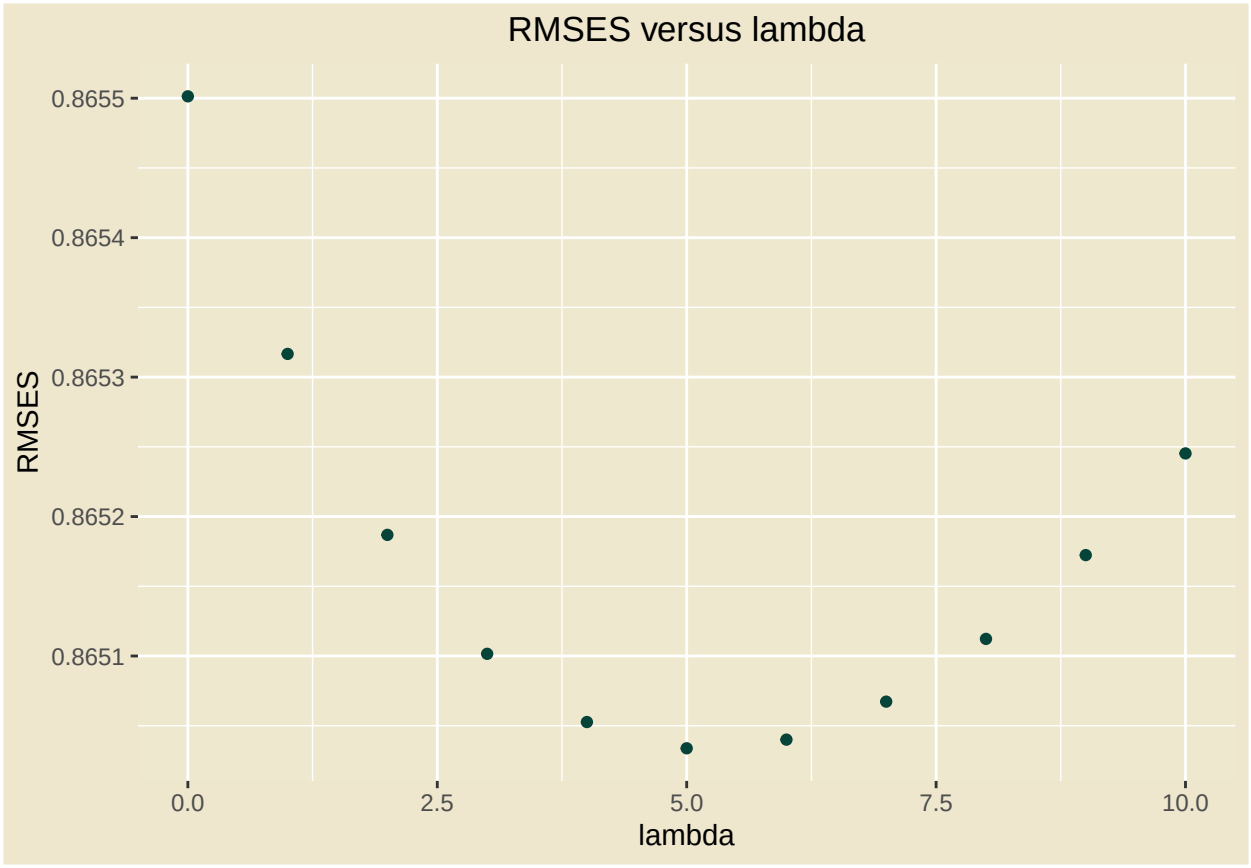
```
movie_reg <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - edx_average)/(n()+lambda[which.min(RMSES)]))
```

4.8 Movie + user effect with regularization

```
lambda <- seq(0,10,1)

RMSES <- sapply(lambda, function(l){
  user_reg <- edx_train %>%
    left_join(movie_reg, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - edx_average - b_i)/(n()+l))

  predicted_ratings <-  edx_test %>%
    left_join(user_reg, by="userId") %>%
    left_join(movie_reg, by="movieId") %>%
    mutate(pred = edx_average + b_i + b_u ) %>% pull(pred)

  return(RMSE(predicted_ratings, edx_test$rating))
})
user_reg_m2 <- min(RMSES)

lambda[which.min(RMSES)]
```

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

Plot RMSES versus lambda



Results:

```
kable(tibble(Model_Type = c("mean base model", "Movie effect model",
                            "Movie + user effect model",
                            "Movie + user + year effect model",
                            "movie + user + year + rate date effect",
                            "movie + user + year + rate date + genre effect"),
             RMSE = c(mean_m1, movie_m2, movie_user_m3, movie_user_year_m4,
                      movie_user_year_ratedate_m5, movie_user_year_ratedate_genres_m6),
             RMSE_with_reg=c(NA,movie_reg_m2, user_reg_m2,NA,NA,NA)) %>%
                    mutate(RMSE = sprintf("%0.6f", RMSE),
                           RMSE_with_reg = sprintf("%0.6f", RMSE_with_reg)))
```

Model_Type	RMSE	RMSE_with_reg
mean base model	1.060333	NA
Movie effect model	0.943626	0.943566
Movie + user effect model	0.865629	0.865034
Movie + user + year effect model	0.865309	NA
movie + user + year + rate date effect	0.865224	NA
movie + user + year + rate date + genre effect	0.865131	NA

Redefine user + movie effect

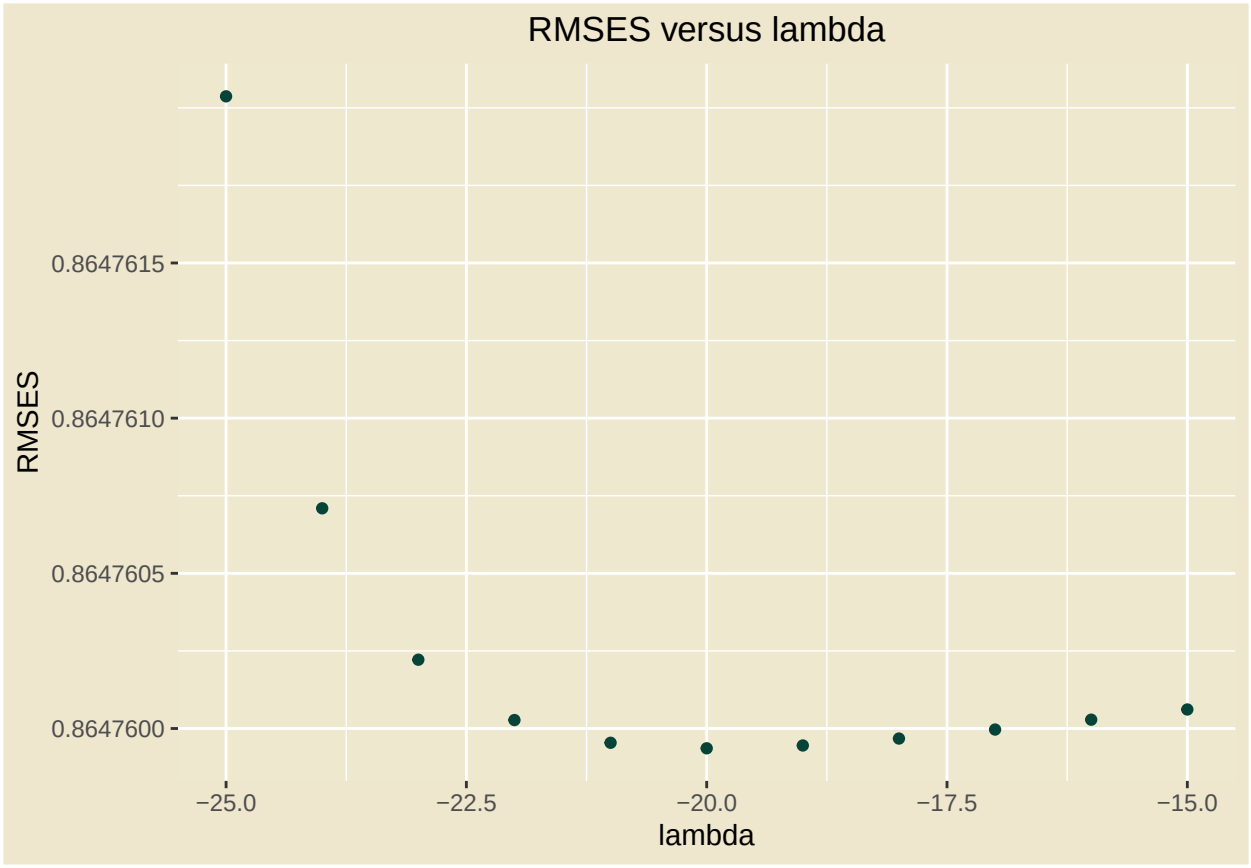
```
user_reg <- edx_train %>%
    left_join(movie_reg, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - edx_average - b_i)/(n()+lambda[which.min(RMSES)]))
```

4.9 Movie + user + year effect with regularization

```
lambda <- seq(-25,-15,1)

RMSES <- sapply(lambda, function(l){
  year_reg <- edx_train %>%
    left_join(movie_reg, by="movieId") %>%
    left_join(user_reg, by = "userId") %>%
    group_by(year_of_prod) %>%
    summarize(b_a = sum(rating - edx_average - b_i - b_u)/(n()+l))

  predicted_ratings <-  edx_test %>%
    left_join(user_reg, by="userId") %>%
    left_join(movie_reg, by="movieId") %>%
    left_join(year_reg, by="year_of_prod") %>%
    mutate(pred = edx_average + b_i + b_u + b_a) %>% pull(pred)

  return(RMSE(predicted_ratings, edx_test$rating))
})

year_reg_m3 <- min(RMSES)

lambda[which.min(RMSES)]
```

```
## [1] -20
```

Plot RMSES versus lambda



Results:

```
kable(tibble(Model_Type = c("mean base model", "Movie effects model",
                            "Movie + user effects model",
                            "Movie + user + year effects model",
                            "movie + user + year + rate date effect",
                            "movie + user + year + rate date + genre effect"),
             RMSE = c(mean_m1, movie_m2, movie_user_m3, movie_user_year_m4,
                      movie_user_year_ratedate_m5, movie_user_year_ratedate_genres_m6),
             RMSE_with_reg=c(NA,movie_reg_m2, user_reg_m2, year_reg_m3,NA,NA)) %>%
                    mutate(RMSE = sprintf("%0.6f", RMSE),
                           RMSE_with_reg = sprintf("%0.6f", RMSE_with_reg)))
```

Model_Type	RMSE	RMSE_with_reg
mean base model	1.060333	NA
Movie effects model	0.943626	0.943566
Movie + user effects model	0.865629	0.865034
Movie + user + year effects model	0.865309	0.864760
movie + user + year + rate date effect	0.865224	NA
movie + user + year + rate date + genre effect	0.865131	NA

Redefine movie + user + year effect

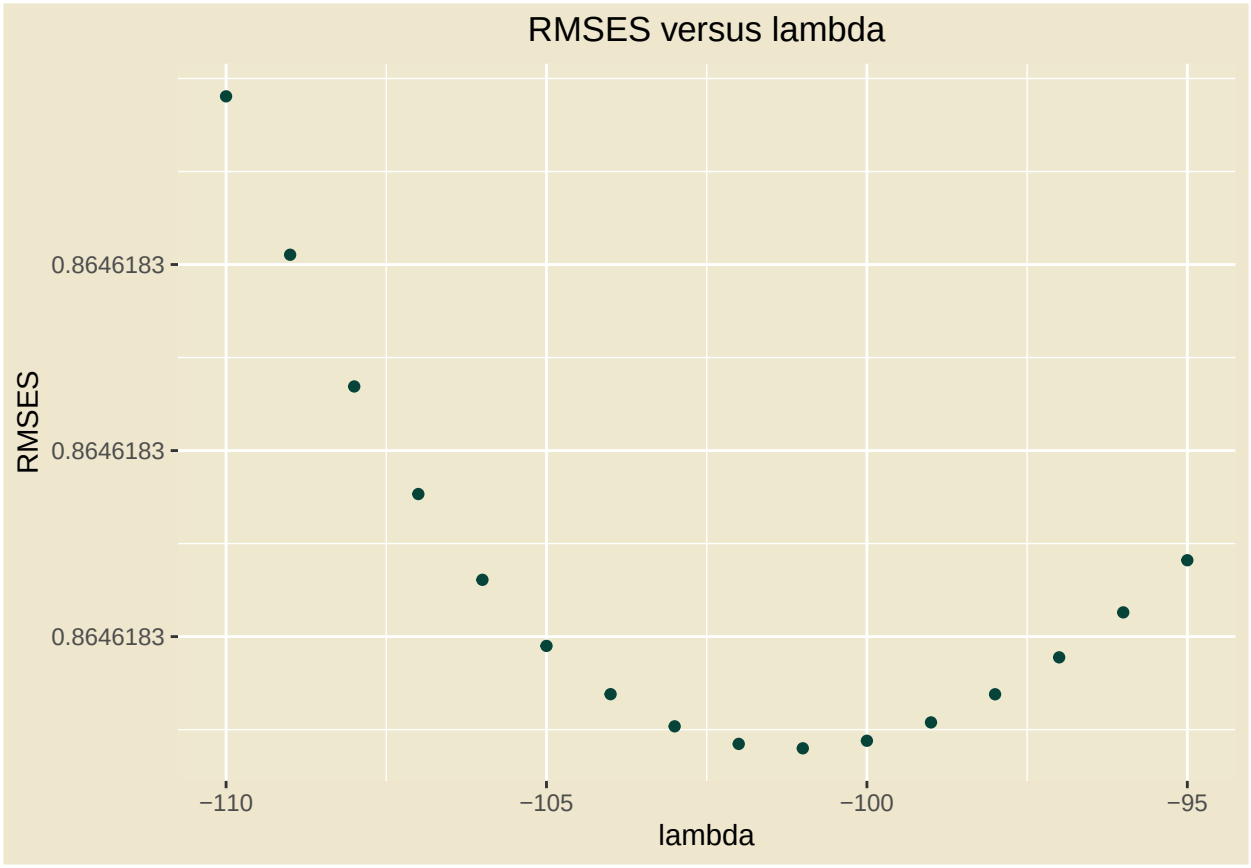
```
year_reg <- edx_train %>%
    left_join(movie_reg, by="movieId") %>%
    left_join(user_reg, by = "userId") %>%
    group_by(year_of_prod) %>%
    summarize(b_a = sum(rating - edx_average - b_i - b_u)/(n()+lambda[which.min(RMSES)]))
```

4.10 Movie + user + year + rate_date effect with regularization

```
lambda <- seq(-110,-95,01)

RMSES <- sapply(lambda, function(l){
  rate_date_reg <- edx_train %>%
    left_join(movie_reg, by="movieId") %>%
    left_join(user_reg, by = "userId") %>%
    left_join(year_reg, by="year_of_prod") %>%
    group_by(rate_date) %>%
    summarize(d = sum(rating - edx_average - b_i - b_u - b_a)/(n()+l))

  predicted_ratings <-  edx_test %>%
    left_join(user_reg, by="userId") %>%
    left_join(movie_reg, by="movieId") %>%
    left_join(year_reg, by="year_of_prod") %>%
    left_join(rate_date_reg, by="rate_date") %>%
    mutate(pred = edx_average + b_i + b_u + b_a +d) %>% pull(pred)

  return(RMSE(predicted_ratings, edx_test$rating))
})
rate_date_reg_m4 <- min(RMSES)

lambda[which.min(RMSES)]
```

```
## [1] -101
```

Plot RMSES versus lambda



Results:

```
kable(tibble(Model_Type = c("mean base model", "Movie effect model",
                            "Movie + user effect model",
                            "Movie + user + year effect model",
                            "movie + user + year + rate date effect",
                            "movie + user + year + rate date + genre effect"),
             RMSE = c(mean_m1, movie_m2, movie_user_m3, movie_user_year_m4,
                      movie_user_year_ratedate_m5, movie_user_year_ratedate_genres_m6),
             RMSE_with_reg=c(NA,movie_reg_m2, user_reg_m2, year_reg_m3,
                             rate_date_reg_m4, NA)) %>%
                      mutate(RMSE = sprintf("%0.6f", RMSE),
                             RMSE_with_reg = sprintf("%0.6f", RMSE_with_reg)))
```

Model_Type	RMSE	RMSE_with_reg
mean base model	1.060333	NA
Movie effect model	0.943626	0.943566
Movie + user effect model	0.865629	0.865034
Movie + user + year effect model	0.865309	0.864760
movie + user + year + rate date effect	0.865224	0.864618
movie + user + year + rate date + genre effect	0.865131	NA

Redefine movie + user + year + rate_date effect

```
rate_date_reg <- edx_train %>%
    left_join(movie_reg, by="movieId") %>%
    left_join(user_reg, by = "userId") %>%
    left_join(year_reg, by="year_of_prod") %>%
    group_by(rate_date) %>%
    summarize(d = sum(rating - edx_average - b_i - b_u - b_a)/(n()+lambda[which.min(RMSES)]))
```

5. Final Results

To calculate final RMSE use final_holdout_test which wasn't used in earlier calculation as test set. To do this use model which achieved the best result - movie + user + year + rate_date effect with regularization

Create columns year_of_prod and rate_date in final_holdout_test same as in edx set.

```
final_holdout_test <- final_holdout_test %>%
  mutate(year_of_prod = str_extract(title, pattern= ".\\d{4}.$")) %>%
  mutate(year_of_prod = str_extract(year_of_prod, pattern = "\\d{4}"))

final_holdout_test <- final_holdout_test %>% mutate(rate_date = as_datetime(timestamp))
final_holdout_test$rate_date <- round_date(final_holdout_test$rate_date, unit="month")
```

Make sure movieId and userId in final_holdout_test set are also in edx_train set

```
final_holdout_test <- final_holdout_test %>%
  semi_join(edx_train, by = "movieId") %>%
  semi_join(edx_train, by = "userId")
```

Predict ratings in final_holdout_test set and calculate RMSE

```
final_predictions <- final_holdout_test %>%
  left_join(user_reg, by="userId") %>%
  left_join(movie_reg, by="movieId") %>%
  left_join(year_reg, by="year_of_prod") %>%
  left_join(rate_date_reg, by="rate_date") %>%
  mutate(pred = edx_average + b_i + b_u + b_a +d) %>% pull(pred)
final_m <- RMSE(final_predictions, final_holdout_test$rating)
```

Results:

```
kable(tibble(Model_Type = c("mean base model", "Movie effect model",
                            "Movie + user effect model",
                            "Movie + user + year effects model",
                            "movie + user + year + rate date effect",
                            "movie + user + year + rate date + genre effect"),
             RMSE = c(mean_m1, movie_m2, movie_user_m3, movie_user_year_m4,
                              movie_user_year_ratedate_m5,
                              movie_user_year_ratedate_genres_m6),
             RMSE_with_reg = c(NA,movie_reg_m2, user_reg_m2, year_reg_m3,
                                      rate_date_reg_m4, NA),
             Final_prediction =c (NA,NA,NA,NA,final_m,NA)) %>%
                      mutate(RMSE = sprintf("%0.6f", RMSE),
```

```
                                RMSE_with_reg = sprintf("%0.6f", RMSE_with_reg),
                                Final_prediction = sprintf("%0.6f", Final_prediction)))
```

Model_Type	RMSE	RMSE_with_reg	Final_prediction
mean base model	1.060333	NA	NA
Movie effect model	0.943626	0.943566	NA
Movie + user effect model	0.865629	0.865034	NA
Movie + user + year effects model	0.865309	0.864760	NA
movie + user + year + rate date effect	0.865224	0.864618	0.864752
movie + user + year + rate date + genre effect	0.865131	NA	NA

6. Conclusion

Report contains data preparation, data analysis and models creating. Data analysis shows characteristic of data and allows better understand how can be model algorithm for this project. In ext section were created few models and displayed RMSE results they received. The best result received model movie + user + year + rate date effect with regularization which showed RMSE 0.86475 on predicting final holdout test. It means that the goal of this project(RMSE <0.86490) has been achieved. Modeling process can be developing by adding to best model genre effect with regularization or using machine-learning algorithms as “glm” or “knn”.