MovieLens

Daoud Youssef

2020-06-22

Contents

Introduction **1**

Methodology **2**
Dataset . . . 2

Data analysis and model preparation **3**
Data exploration and visualization . . . 3
Modeling . . . 14
Model 1: Average movie rating . . . 14
Model 2: Movie effect model . . . 15
Model 3: Movie and user effect model . . . 15
Model 4: Regularized movie and user effect model. . . . 15
Validating the model . . . 17

Results **17**
Parameter λ . . . 17
RMSE Results . . . 19

Conclusion and discussion **19**

References **20**

Introduction

This project is part of the Harvard: PH125.9 course Data science: Capstone course. It consists of building a recommender system to predict the rating of a movie given by a user.

Originally, Netflix awarded a $1M Grand Prize to improve by 10% the accuracy of rating predictions.[1]

The aim of this project is to buid an algorithm to predict the rating of a movie. The evaluation of the validity of our model is based on the Residual Mean Squared Error RMSE. The target is to reach a value below 0.9. This error is interpreted as movie rating and thus an error of value 1 means an error of 1 star in predicting. (Irizarry, n.d.)

This project is organized as follows: In section 2, we present the dataset used and detailed explanation on all of the models.In section 3, we explore the `Edx` dataset and its statistical properties. In addition we present the algorithms of the models used in this project. In Section 4, we discuss the results and we conclude in section 5.

[1] https://www.netflixprize.com

Methodology

Dataset

First, we load libraries and data. This code is provided by the `Edx` team.[2]. This code stopped running and alernatevily, I used downloaded packages for data.

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

```
if (!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if (!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if (!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

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

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.ch

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

```
## Code provided above from edx stopped working so I used a file saved before
edx <- readRDS("edx.rds")
validation <- readRDS("validation.rds")
```

There is 2 sets used in this project :

1. **Edx** set which is used to analyze the predictors and build our model. This data set has 9000055 rows and 6 columns. This data set is partitioned into two sets `train_edx` and `test_edx`. The `train_set` represents 90% of the `edx` data.
2. **Validation** set is used to evaluate and validate our final model. This dataset has 999999 and 6 columns. As per the project guidelines, the validation set will be 10% of the MovieLens data.

We will use linear regression model to predict the rating of the movie based on our variables. The goal is to examine how strong is the relationship between the outcome and predictors.[3]

For this project, we build 3 models. First model, baseline model, we used the mean of ratings as predictor. Second model, we add another the movie effect as a new parameter. Third model, the user effect was added to the previous model.

To optimize our model, we regularized it by introducing a new parameter. The goal is always to minimize the sum of the squared residuals and give our model more stability.

Validation of our model is based on the value of Root Mean Square Error **RMSE**. The **RMSE** loss function is calculated as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}\left(y_{u,i} - \hat{y}_{u,i}\right)^2} \tag{1}$$

with

[2]https://courses.edx.org/courses/course-v1:HarvardX+PH125.9x+1T2020/courseware/dd9a048b16ca477a8f0aaf1d888f0734/e8800e37aa444297a3a2f35bf84ce452/?activate_block_id=block-v1%3AHarvardX%2BPH125.9x%2B1T2020%2Btype%40sequential%2Bblock%40e8800e37aa444297a3a2f35bf84ce452

[3]search for it

- $y_{u,i}$ the actual ratings,
- $\hat{y}_{u,i}$ the predicted ratings.
- N number of possible combinations between user u and movie i.

The code that compute the RMSE is :

```
RMSE <- function(true_ratings, predicted_ratings) {
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

Finally, we use the validation set to test our model.

Data analysis and model preparation

Data exploration and visualization

First, the `edx` dataset contains no missing values.

```
anyNA(edx)
```

```
## [1] FALSE
```

The `edx` dataset contains 6 variables. `UserId` and `movieId` are our predictors and `rating` is the outcome.

```
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 292 316 329 355 356 362 364 370 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983421 838983392 838983392 838984474 838983653 838984885 83
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargate (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|Thriller" "Action|Ac
```

```
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
```

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

From the code below, we can conclude that there is no zero rating for any movie,

```
sum(edx$rating == 0)
```

```
## [1] 0
```

```
user <- edx %>% summarize(n_users = n_distinct(userId), n_movies = n_distinct(movieId))
knitr::kable(user, col.names = c("Number of users", "Number of Movies"))
```

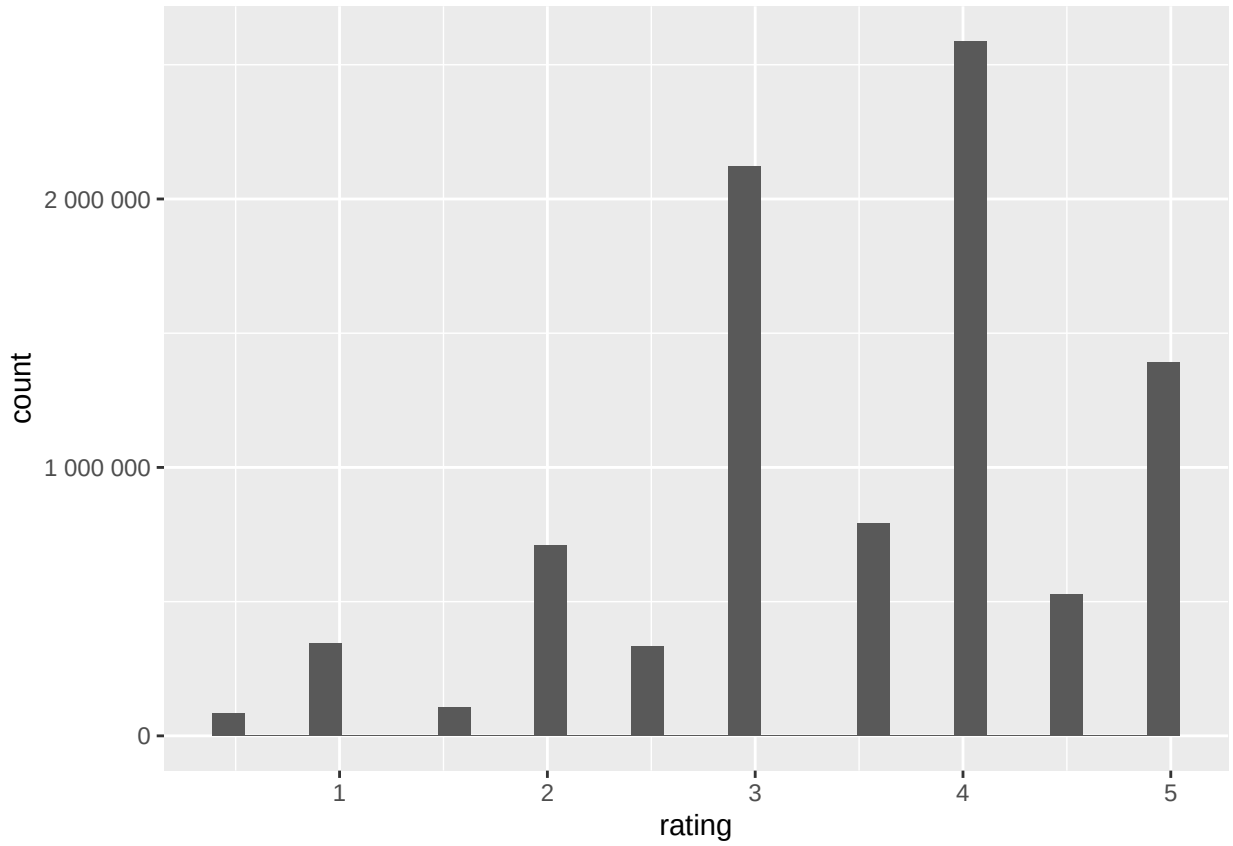
Number of users	Number of Movies
69878	10677

Edx data set has 69878 different users and 10677 different movies. Therefore, Some users are rating more than one movie.

```
rate_distribution <- edx %>%
  group_by(rating) %>%
  summarize(n = n())
rate_distribution %>% knitr::kable(col.names = c("Rating", "Number of rates"))
```

Rating	Number of rates
0.5	85374
1.0	345679
1.5	106426
2.0	711422
2.5	333010
3.0	2121240
3.5	791624
4.0	2588430
4.5	526736
5.0	1390114

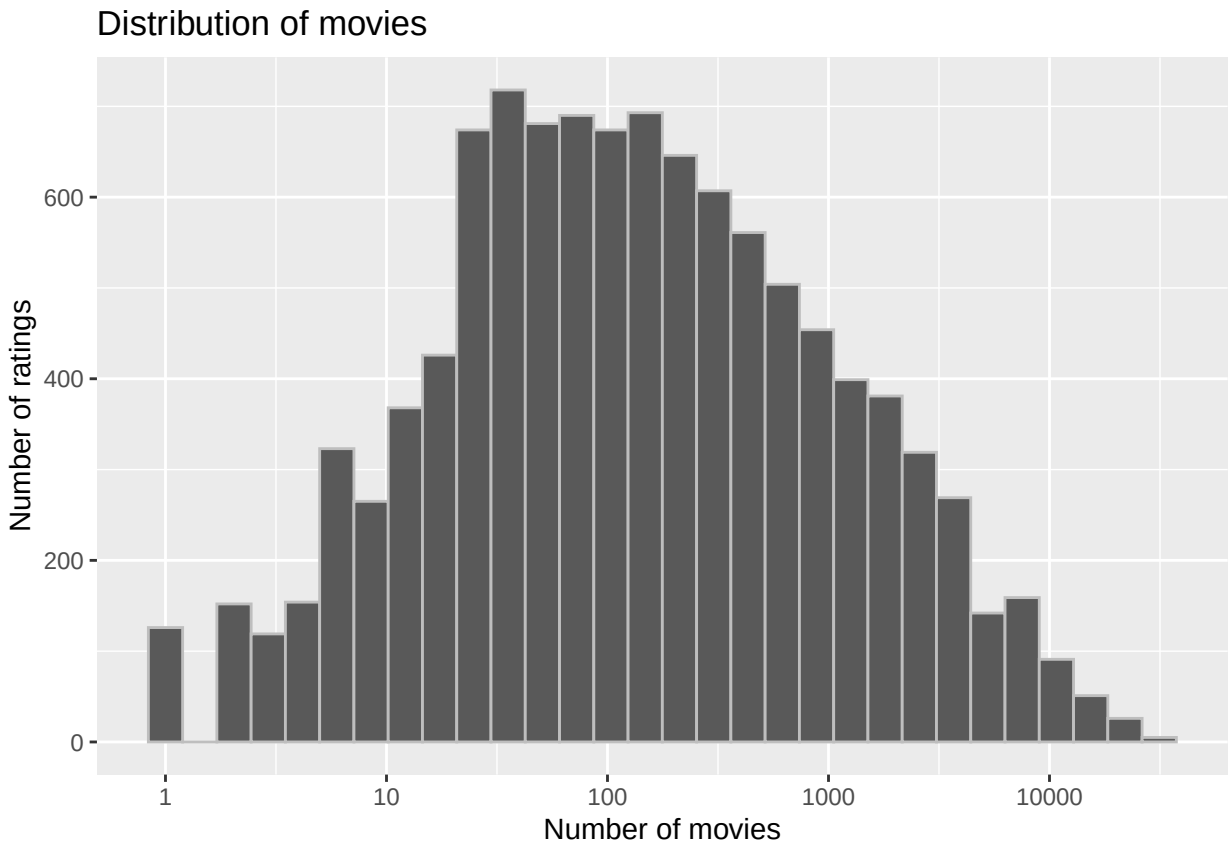
```
edx %>%
  group_by(rating) %>%
  ggplot(aes(x = rating)) +
  geom_histogram(bins = 30) +
  scale_y_continuous(labels = function(x) format(x, big.mark = " ", scientific = FALSE))
```



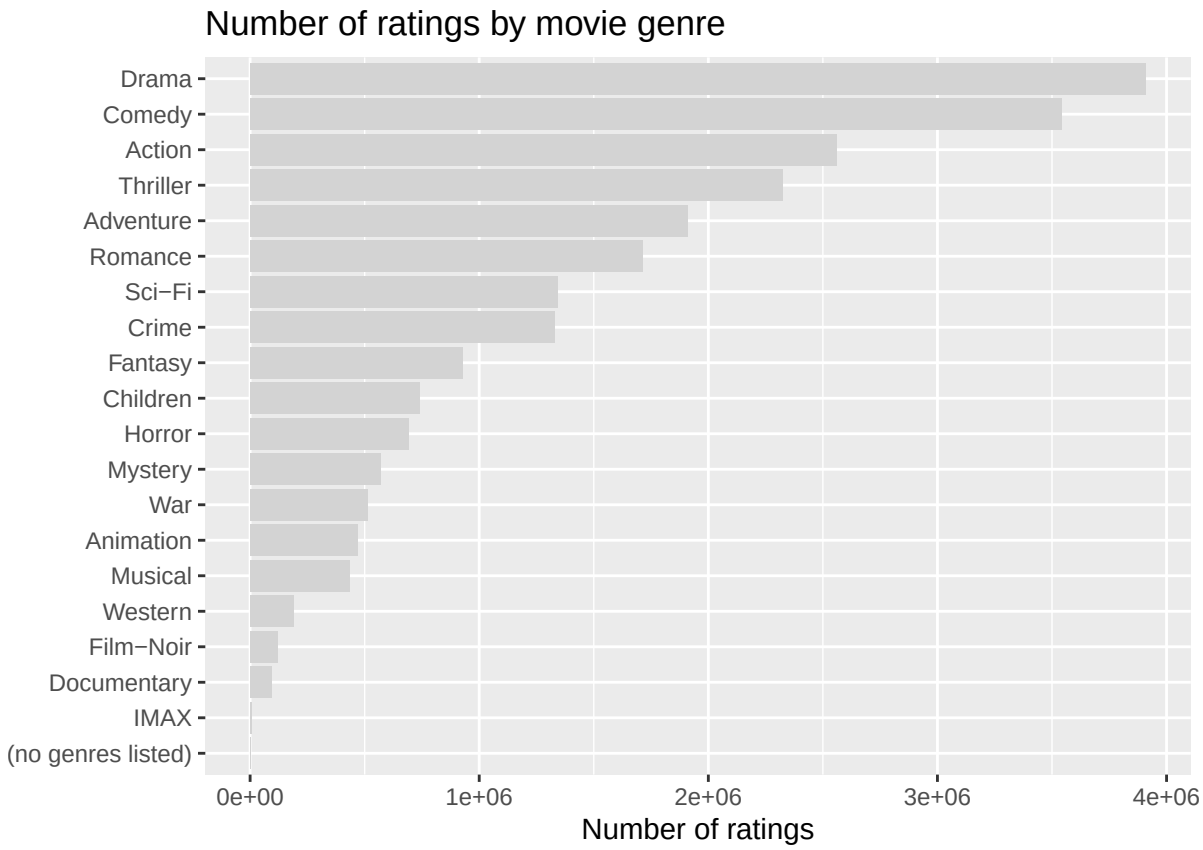
We notice from the rating distribution the following:

1. Half star ratings are less rated than whole star ratings.
2. The 4 star rating is the most applied among other ratings.

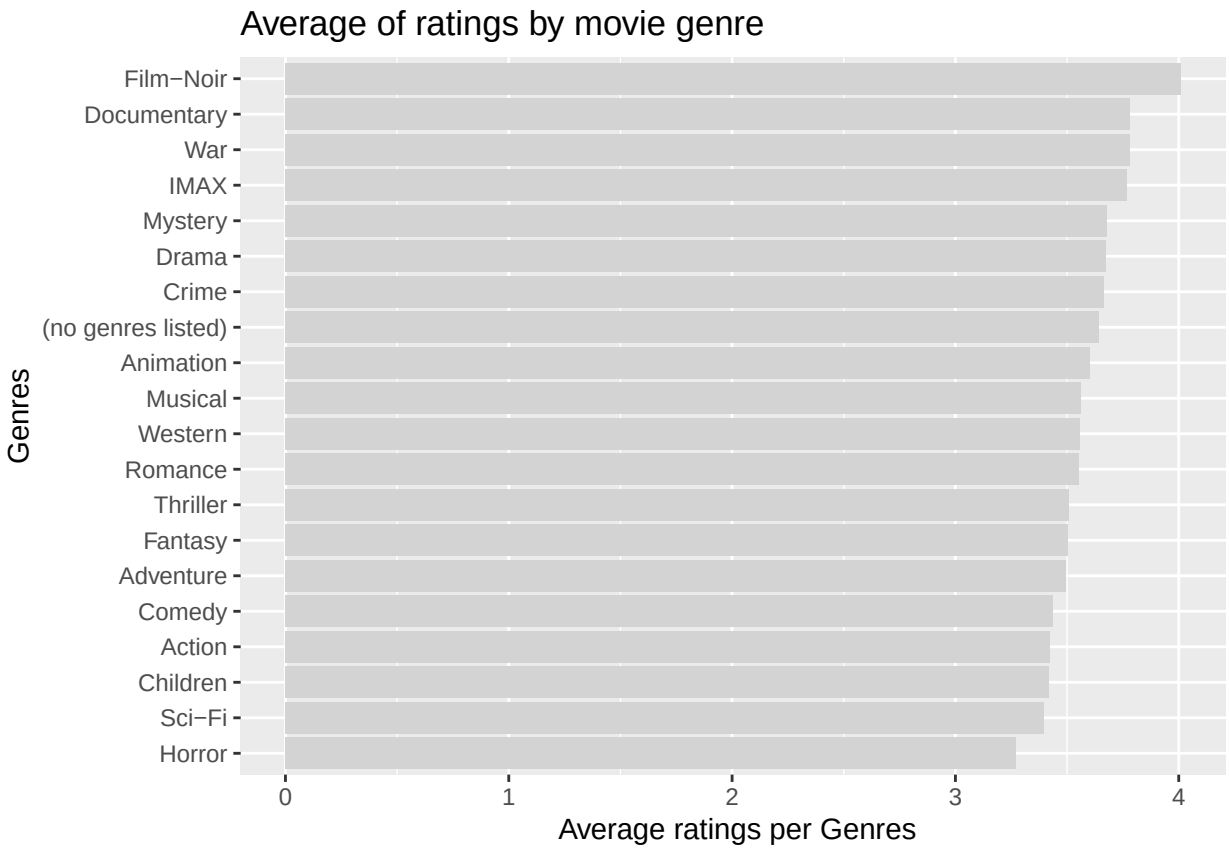
```
edx %>%
  group_by(movieId) %>%
  summarize(number_movies = n()) %>%
  ggplot(aes(number_movies)) +
  geom_histogram(bins = 30, color = "grey") +
  scale_x_log10() +
  ggtitle("Distribution of movies") +
  xlab("Number of movies") +
  ylab("Number of ratings")
```




```
genre <- edx %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(count = n()) %>%
  arrange(desc(count))
genre %>% ggplot(aes(x = reorder(genres, count), y = count)) +
  geom_bar(stat = "identity", fill = "lightgrey") +
  labs(x = "", y = "Number of ratings") +
  coord_flip() +
  labs(title = "Number of ratings by movie genre")
```



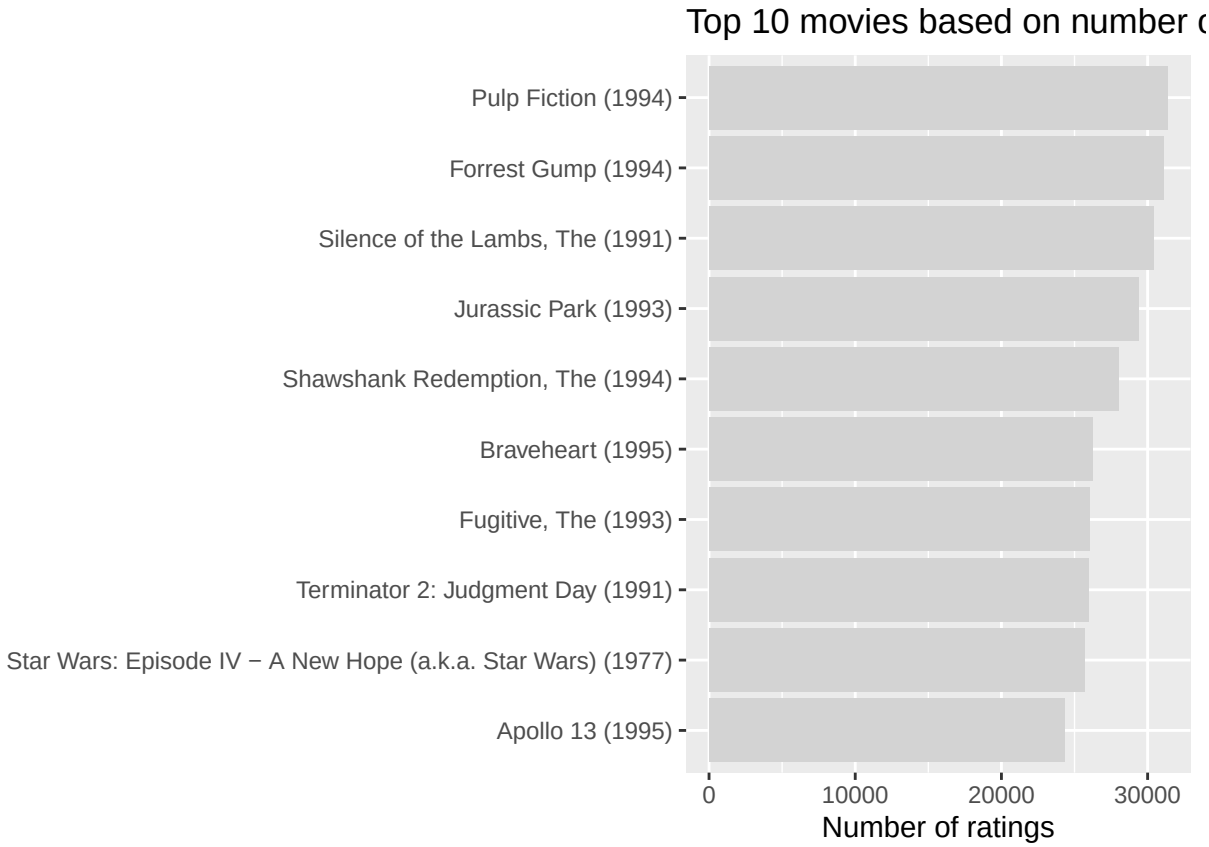

```
genre_rating <- edx %>%
  separate_rows(genres, sep = "\\|") %>%
  group_by(genres) %>%
  summarize(n = mean(rating)) %>%
  arrange(desc(n))

genre_rating %>% ggplot(aes(x = n, y = reorder(genres, n))) +
  geom_bar(stat = "identity", fill = "lightgrey") +
  labs(x = " Average ratings per Genres", y = "Genres") +
  labs(title = "Average of ratings by movie genre")
```



From the chart above, we conclude that Drama is the top genre rated according to users, followed by Comedy and action. We note that 7 movies are not been categorized by any genre. Film-noir genre has a high rating level but less number of ratings. Drama is more rated than others but as average rating is not high.

The 10 most rated movies are shown below. 5 of these movies are categorized as Drama and 4 in the action category.

```
top_10 <- edx %>%
  group_by(title, genres) %>%
  summarize(n = n()) %>%
  arrange(desc(n))

top_10[1:10, ] %>% ggplot(aes(x = reorder(title, n), y = n)) +
  geom_bar(stat = "identity", fill = "lightgrey") +
  labs(x = "", y = "Number of ratings") +
  coord_flip() +
  labs(title = "Top 10 movies based on number of ratings")
```




```
ratebygenre <- edx %>%
  group_by(genres) %>%
  summarize(n = n(), mean_rate = mean(rating)) %>%
  arrange(desc(n))
ratebygenre[1:10, ] %>% knitr::kable(col.names = c("Genre", "Number of ratings", "Average of ratings"))
```

Genre	Number of ratings	Average of ratings
Drama	733296	3.712364
Comedy	700889	3.237858
Comedy|Romance	365468	3.414486
Comedy|Drama	323637	3.598961
Comedy|Drama|Romance	261425	3.645824
Drama|Romance	259355	3.605471
Action|Adventure|Sci-Fi	219938	3.507407
Action|Adventure|Thriller	149091	3.434101
Drama|Thriller	145373	3.446345
Crime|Drama	137387	3.947135

We can conclude that top 10 movies as rating number does not have a higher average in rating.

```
edx$timestamp <- as_datetime(as.POSIXct(edx$timestamp, origin = "1970-01-01"))
edx_d <- edx %>% mutate(days = weekdays(edx$timestamp), monthss = months(edx$timestamp), years = year(ed
edx_d$days <- factor(edx_d$days, levels = c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Sa
edx_d$monthss <- factor(edx_d$monthss, levels = c("January", "February", "March", "April", "May", "June
edx_d %>%
  group_by(monthss) %>%
```

```
  summarize(n = n()) %>%
  arrange(desc(n)) %>%
  knitr::kable(col.names = c("Month", "Number of ratings")) ## rating distribution based on months
```

Month	Number of ratings
November	975513
December	900548
October	829510
July	802607
January	752413
June	748575
March	744315
August	724498
April	671145
May	665393
February	618184
September	567354

```
edx_d %>%
  group_by(days) %>%
  summarize(n = n()) %>%
  arrange(desc(n)) %>%
  knitr::kable(col.names = c("Days", "Number of ratings")) ## rating
```

Days	Number of ratings
Tuesday	1431735
Monday	1410155
Wednesday	1332505
Thursday	1241493
Friday	1232250
Sunday	1226472
Saturday	1125445

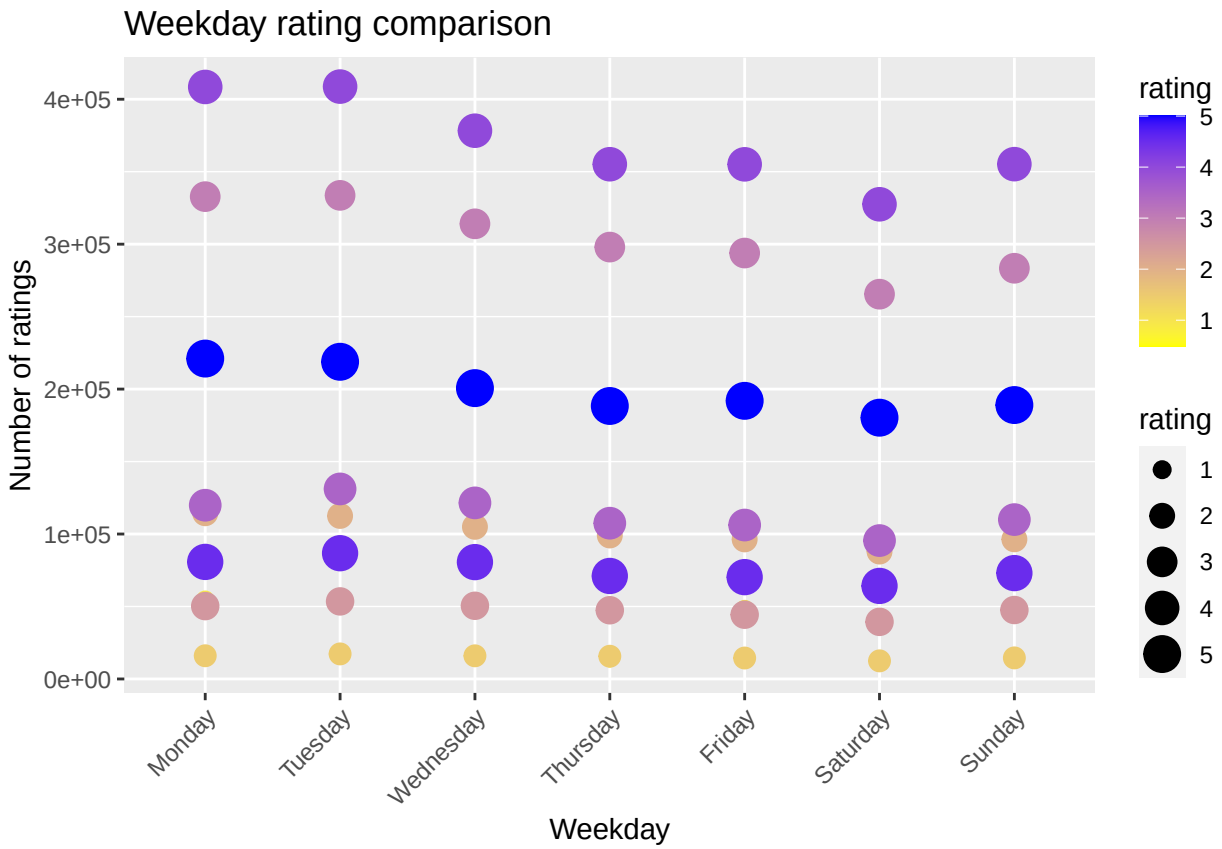
```
edx_d %>%
  group_by(years) %>%
  summarize(n = n()) %>%
  arrange(desc(n)) %>%
  knitr::kable(col.names = c("Years", "Number of ratings"))
```

Years	Number of ratings
2000	1144349
2005	1059277
1996	942772
1999	709893
2008	696740
2004	691429
2006	689315
2001	683355
2007	629168
2003	619938

Years	Number of ratings
2002	524959
1997	414101
1998	181634
2009	13123
1995	2

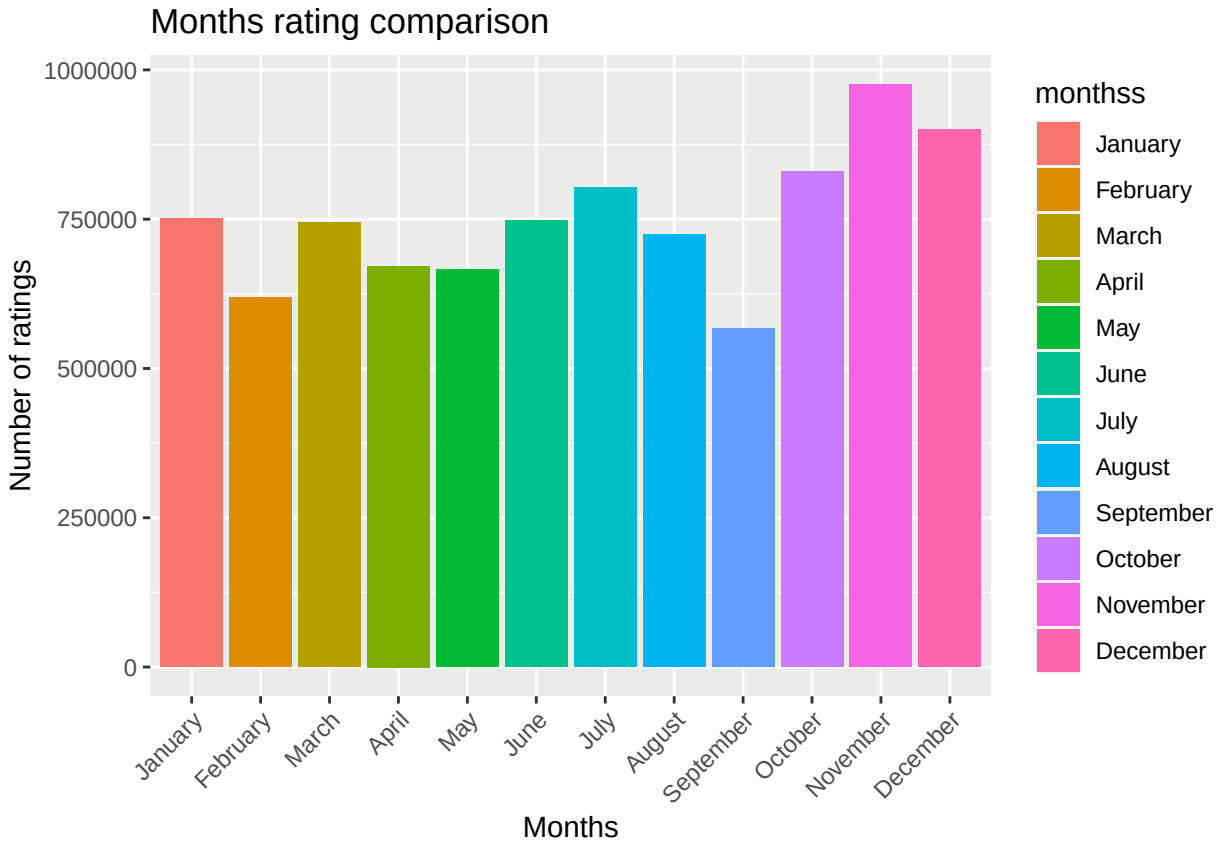
Number ratings is high during November. Tuesday seems the day where most users rate. 2000 is the year where most number of ratings had been done.

```
edx_d %>%
  group_by(days, rating) %>%
  summarize(n = n()) %>%
  ggplot(aes(days, n)) +
  geom_point(stat = "identity", aes(color = rating, size = rating)) +
  scale_color_gradient(low = "yellow", high = "blue") +
  labs(x = "Weekday", y = "Number of ratings", title = "Weekday rating comparison")+ theme(axis.text.x =
```



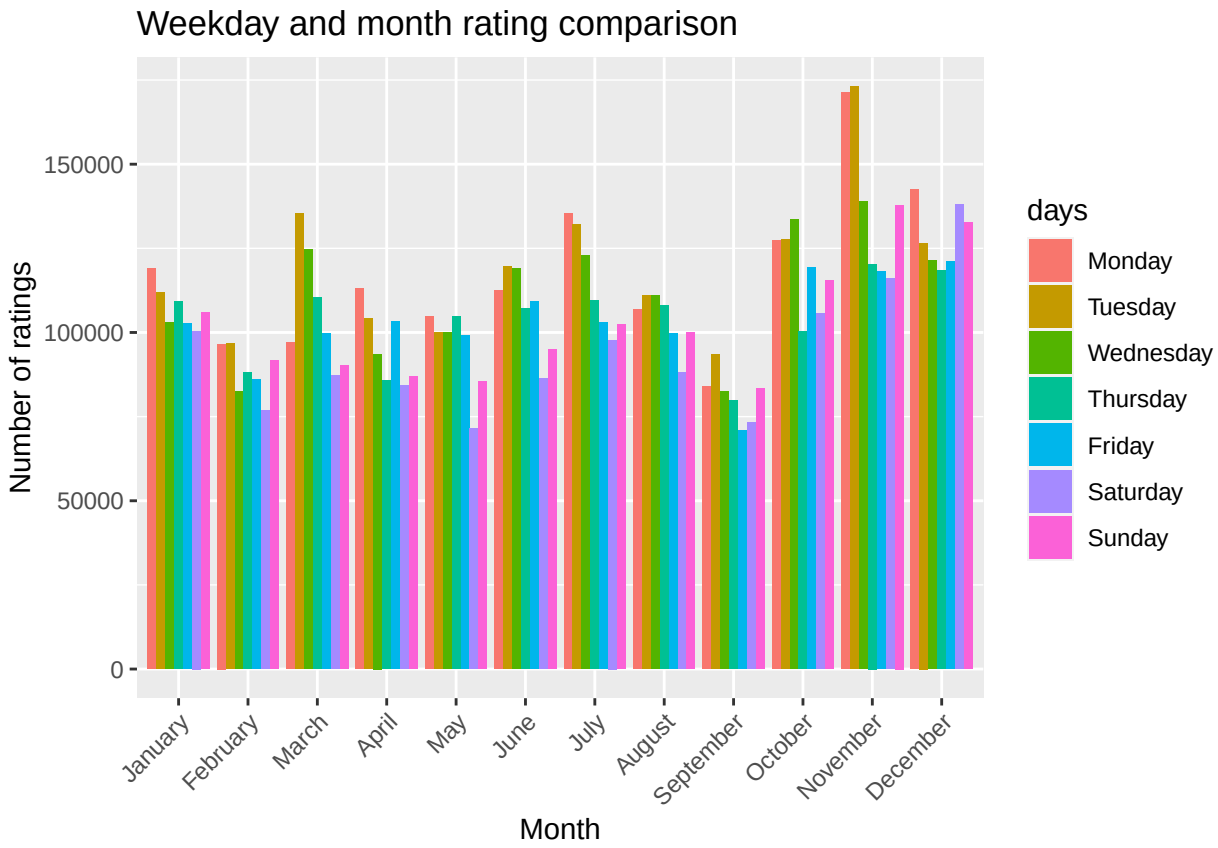
As shown by graph above, weekdays seems does not have a strong influence on the ratings of the movies.

```
edx_d %>%
  group_by(monthss, rating) %>%
  summarize(n = n()) %>%
  ggplot(aes(monthss, n, fill = monthss)) +
  geom_bar(stat = "identity") +
  labs(x = "Months", y = "Number of ratings", title = "Months rating comparison")+ theme(axis.text.x = 
```

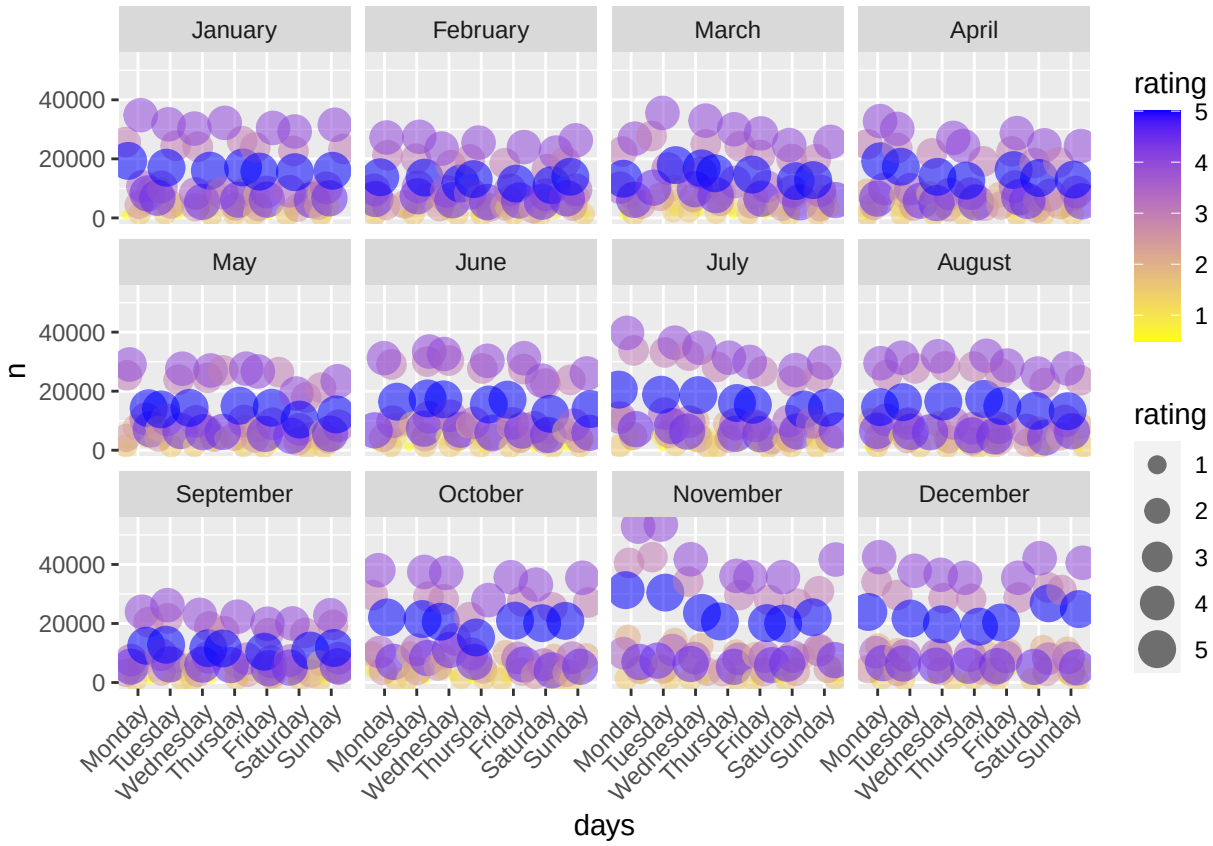


November and December have the highest number of rating respectively.

```
edx_d %>%
  group_by(days, monthss) %>%
  summarize(n = n()) %>%
  ggplot(aes(monthss, n)) +
  geom_bar(stat = "identity", aes(fill = days), position = "dodge") +
  labs(x = "Month", y = "Number of ratings", title = "Weekday and month rating comparison")+ theme(axis
```




```
edx_d %>%
  group_by(days, monthss, rating) %>%
  summarize(n = n()) %>%
  ggplot(aes(days, n)) +
  geom_jitter(aes(color = rating, size = rating), alpha = 0.55) +
  scale_color_gradient(low = "yellow", high = "blue") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  facet_wrap(vars(monthss))
```



Modeling

As we mentioned above, we excluded 10% of `Edx` dataset for testing and we keep 90% for training.

```
# edx data set will splited into training (90%) and test set.
set.seed(1, sample.kind = "Rounding")
edx_index <- createDataPartition(y = edx$rating, times = 1, p = 0.1, list = FALSE)
train_edx <- edx[-edx_index, ]
temp <- edx[edx_index, ]
## make sure userID and movie Id in test set and in train set
test_edx <- temp %>%
  semi_join(train_edx, by = "movieId") %>%
  semi_join(train_edx, by = "userId")
## add rows from test set back intor train set
removed <- anti_join(temp, test_edx)
train_edx <- rbind(train_edx, removed)
rm(edx_index, temp, removed)
## clean data
train_edx <- train_edx %>% select(userId, movieId, rating, title)
test_edx <- test_edx %>% select(userId, movieId, rating, title)
```

Model 1: Average movie rating

First model is the simplest one. We will be predicting using the average of movie ratings. A rating is approximated as :

$$Y_{u,i} = \mu + \epsilon_{u,i} \tag{2}$$

- μ is the "true" rating of all movies.

- $\epsilon_{u,i}$ is the independent errors sampled from the same distribution centered at 0

```
mu_hat <- mean(train_edx$rating)
baseline_RMSE <- RMSE(test_edx$rating, mu_hat)
RMSE_results <- data.frame(Method = "Average rating movie model", RMSE = baseline_RMSE)
```

Model 2: Movie effect model

With the second model, we add another variable: movie effect. We know that some movies have higher rating than others. We add another term b_i to represent average ranking for movie i. our model become:

$$Y_{u,i} = \mu + b_i + \epsilon_{u,i} \tag{3}$$

```
### Model 2 : adding movie effect  b_i
movie_avgs <- train_edx %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu_hat))
#head(movie_avgs)
# predict the rating with b_v
y_hat_movie_avgs <- mu_hat + test_edx %>%
  left_join(movie_avgs, by = "movieId") %>%
  pull(b_i)
rmse_b_i <- RMSE(test_edx$rating, y_hat_movie_avgs)
## add the results to RMSE results
RMSE_results <- rbind(RMSE_results, data.frame(Method = "Average rating movie + b_i ", RMSE = rmse_b_i)
```

Model 3: Movie and user effect model

Every user rate the movie based on his likes or dislikes and affect the overall rating of the movie. Adding the user effect b_u will improve our model. Therefore, lower RMSE can be achieved. Our model is as follows:

$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i} \tag{4}$$

```
movie_users <- train_edx %>%
  left_join(movie_avgs, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - b_i))
#head(movie_users)
# predict the rating with b_i
y_hat_model <- test_edx %>%
  left_join(movie_avgs, by = "movieId") %>%
  left_join(movie_users, by = "userId") %>%
  mutate(pred = mu_hat + b_i + b_u) %>%
  pull(pred)
rmse_b_i_b_u <- RMSE(test_edx$rating, y_hat_model)
## add the results to RMSE results
RMSE_results <- rbind(RMSE_results, data.frame(Method = "Average rating movie + b_i +b_u", RMSE = rmse_
```

Model 4: Regularized movie and user effect model.

In our algorithms, we are using rating average and RMSE. To prevent overfitting and the total variability of of the effect sizes, we use regularization. It consists of adding a new parameter in order to minimize the loss function. We include λ and tuned it empirically to get the lowest RMSE. We use cross validation method to choose it.

Regularized movie effect model The new model and b_i that minimize the initial model is calculated as follows:

$$\hat{b}_i(\lambda) = \frac{1}{\lambda + N} \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu}) \tag{5}$$

with

- N is the number of ratings for each movie.

```
### regularized movie  effect model
lambdas_e <- seq(0, 10, 0.25)
rmses_e <- sapply(lambdas_e, function(l) {
  mu <- mean(train_edx$rating)
  b_i <- train_edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu) / (n() + l))
  predicted_ratings <- test_edx %>%
    left_join(b_i, by = "movieId") %>%
    mutate(pred = mu + b_i) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_edx$rating))
})
#qplot(lambdas_e, rmses_e)
lambda_e <- lambdas_e[which.min(rmses_e)]
RMSE_results <- rbind(
  RMSE_results,
  data.frame(Method = "Regularized Movie Effect Model", RMSE = min(rmses_e))
)
```

Regularized movie and user effect model The new model and b_i that minimize the initial model is calculated as follows:

$$\hat{b}_i(\lambda) = \frac{1}{\lambda + N} \sum_{u=1}^{n_i} (Y_{u,i} - b_i - \hat{\mu}) \tag{6}$$

with

- N is the number of ratings for each movie.

```
lambdas_u <- seq(0, 10, 0.25)
rmses_u <- sapply(lambdas_u, function(l) {
  mu <- mean(train_edx$rating)
  b_i <- train_edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu) / (n() + l))
  b_u <- train_edx %>%
    left_join(b_i, by = "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu) / (n() + l))
  predicted_ratings <-
    test_edx %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
```

```
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, test_edx$rating))
})
#qplot(lambdas_u, rmses_u)
lambda_u <- lambdas_u[which.min(rmses_u)]
reg_b_i_b_u<-min(rmses_u)
RMSE_results <- rbind(
  RMSE_results,
  data.frame(Method = "Regularized Movie + User Effect Model", RMSE = min(rmses_u))
)
```

Validating the model

We validate our model using the code below. Training set is `Edx` dataset and testing set is the `validation` dataset.

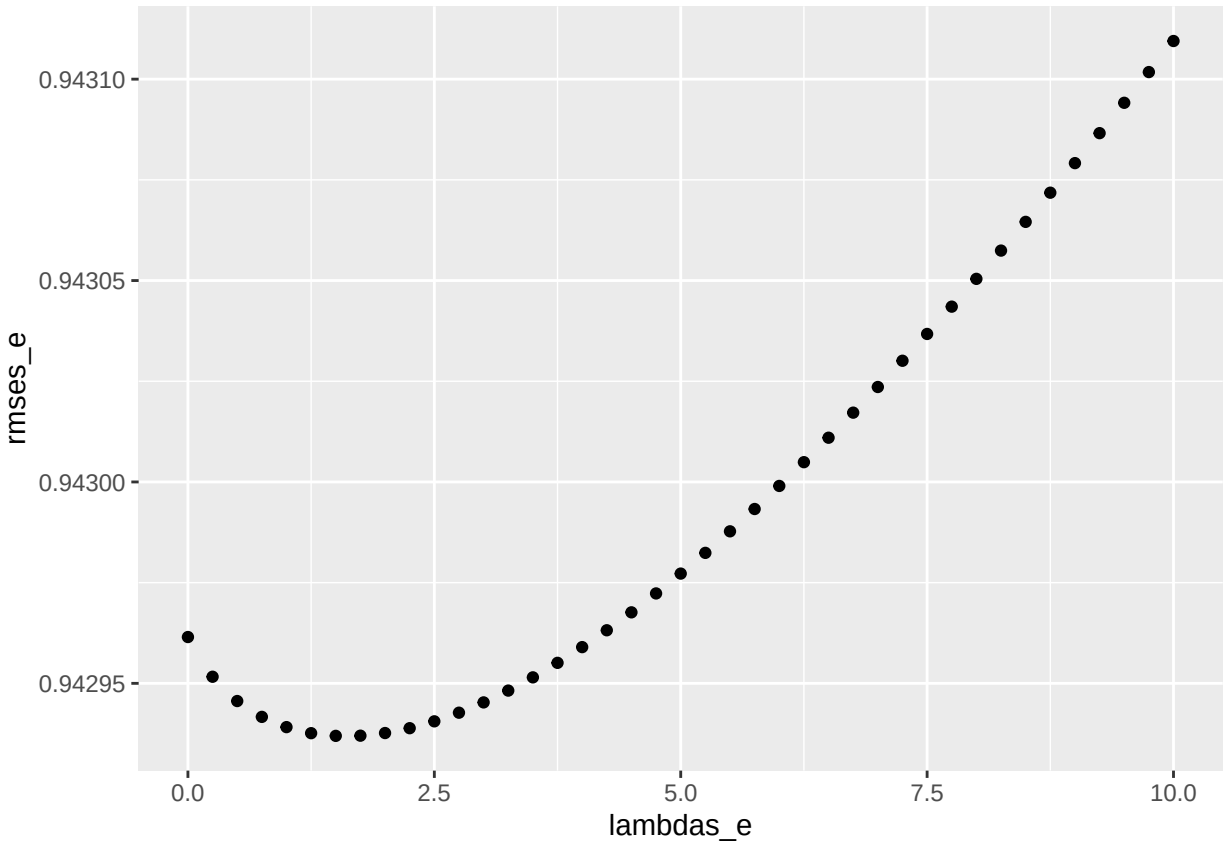
```
mu_edx <- mean(edx$rating)
bi_edx <- edx %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu_edx) / (n() + lambda_u))

bu_edx <- edx %>%
  left_join(bi_edx, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = sum(rating - b_i - mu_edx) / (n() + lambda_u))
predicted_ratings <- validation %>%
  left_join(bi_edx, by = "movieId") %>%
  left_join(bu_edx, by = "userId") %>%
  mutate(pred = mu_edx + b_i + b_u) %>%
  pull(pred)
val_pred <- RMSE(validation$rating, predicted_ratings)


RMSE_results <- rbind(RMSE_results, data.frame(Method = "Validation of edx data set using Movie and Use
```

Results

Parameter λ

For the movie effect model, the optimal λ is :

```
qplot(lambdas_e, rmses_e)
```

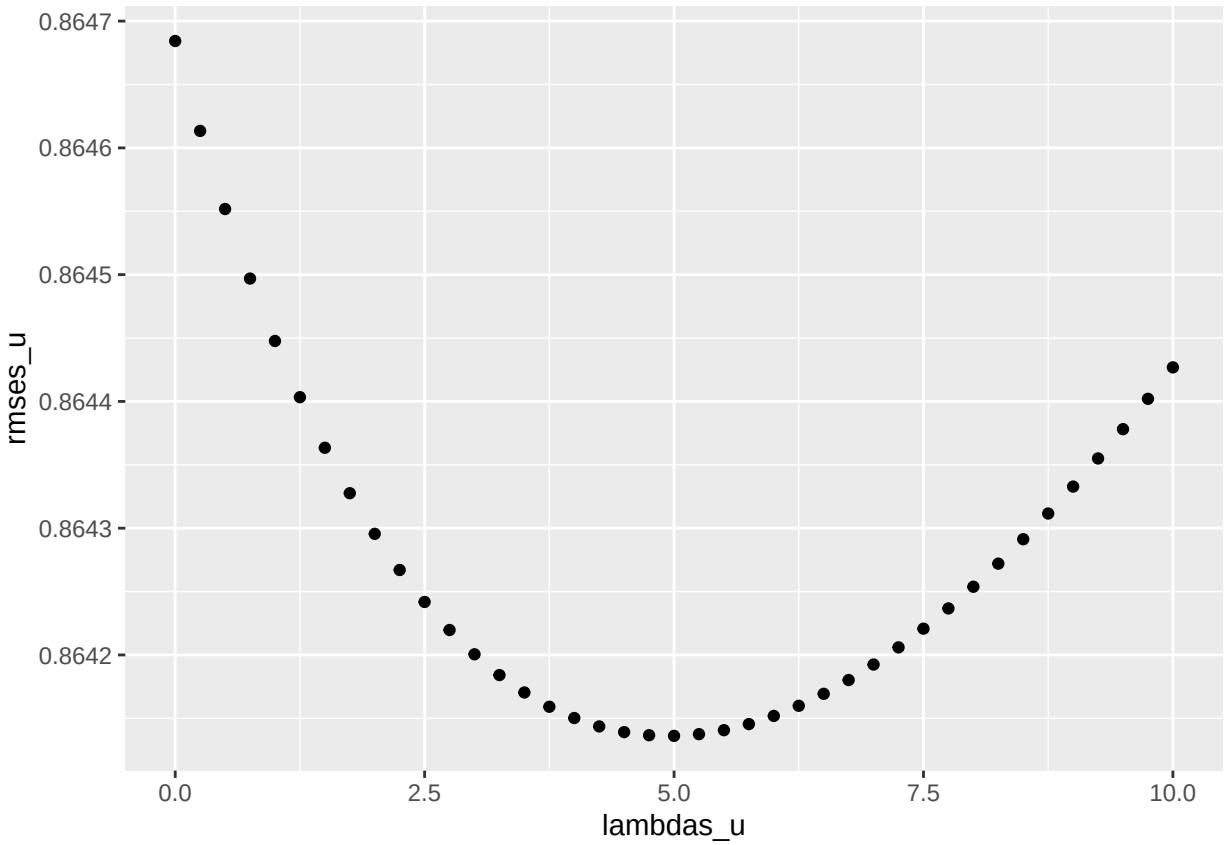



```
lambda_e <- lambdas_e[which.min(rmses_e)]
lambda_e
```

```
## [1] 1.5
```

For the movie and user effect, the optimal λ is :

```
qplot(lambdas_u, rmses_u)
```




```
lambda_u <- lambdas_u[which.min(rmses_u)]
lambda_u
```

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

RMSE Results

Results are shown below:

```
knitr::kable(RMSE_results)
```

Method	RMSE
Average rating movie model	1.0600537
Average rating movie + b_i	0.9429615
Average rating movie + b_i +b_u	0.8646843
Regularized Movie Effect Model	0.9429370
Regularized Movie + User Effect Model	0.8641362
Validation of edx data set using Movie and User Effect Model	0.8648177

The table above shows that the best *RMSE* achieved in the regularized movie and user effect model increased by 0.08% when validating our model.

Conclusion and discussion

This project started by exploring and visualizing the MovieLens data. We built our models starting from the baseline till a more complicated model. The *RMSE* has been reached.

It's interesting to include temporal variability in the predictors for two reasons. First, a user rating may change over time. Second, popularity of movies changes over time as well. A function of time may achieve more consistency in predicting. (example)A low-resolution movie can be rated lower now due to the existing of full HD or 4K movies despite the genre and quality of the movie. Including duration of the movie in the dataset can a play an important parameter in determining the rating of a movie.

References

Irizarry, Rafel A. n.d. *Introduction to Data Science*. https://rafalab.github.io/dsbook/.