Movies Recommendations Capstone Project

Mami Daba Fam

2022-11-05

INTRODUCING MYSELF

My name is Mami Daba Fam, I am a 15 years experienced professional with a proven working in development and humanitarian sector as in industrial sector. I have a double expertise in project management and engineering (R&D, Production, Maintenance) and strong educational background with two master's degree in electromechanical engineering and industrial technologies. I have made my first university studies in the field of mathematics and physics. I am new in the field of data science with a high motivation of challenging a new way to contribute to the achievements of SDGs goals by evidence- based decisions.

My First Capstone project : movies recommendations

1. Overview- Executive summary

This capstone project is the final course of **DATA SCIENCE Professional Certificate Program of HarvardX**. It allows me to apply various skills that was developed in the others eight courses.The main goal is to work independently on a data analysis and machine learning project with minimum guidance from the teacher- Raphael Irizarry.

Recommendation systems are commonly used nowadays. It is a class of machine learning that uses large data sets to recommend products to consumers. Students are asked to work on a movie recommendation systems based on an approach that was explained in *course 9 : Data Science Machine learning*. The MovieLens 10M data set are downloaded on the Grouplens website for the purpose of the project. A validation set approach method will be used to estimate the error rate. The entire data set is divide on a training set named "edx" and a final validation set named "validation".

Different models will be trained on the edx data sets before choosing the model that minimize the RMSE(Root Mean Square Error). This RMSE was used by the Netflix challenge for movies recommendations. Evaluation scores for this capstone project are given by the RMSE achieved with the validation set(25 points/ 100 points). A maximum of 25 points correspond to a **RMSE of 0.86490 and lower.**

The report is structured as following :

-Create training and validation set (the final hold-out test set)

-Explore and visualize the edx data set

-Present the modelling approach and global effects analysis

-Present Results and explain-comment model performance

-Conclusion of the project and future recommendation.

2. Create a training set "edx" and a validation set "validation" (final hold-out test set)

The movielens data set is downloaded on the website and a validation set of 10% is created. This validation set will not be used for model training and optimization.

edx is our training dataset and validation is our final hold-out test set.

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

dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip",
              dl, quiet = TRUE, mode = "wb", method = "auto",
              cacheOK = TRUE, options(timeout = max(1000, getOption("timeout"))))

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")

# if using R 4.0 or later:
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") # if using R 3.5 or earlier, use `set.seed(1)`
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set
validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

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

3. Data exploration

3.1. Structure of edx data set

This exploration step allows us to understand our training data sets and shows roughly the global effects and relationship of our predictors. The data set is in a tidy format and is structured as the following :

```
options(width = 60)
# Structure of edx
str(edx)
```

Classes 'data.table' and '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 838983707 838984596

... $ title : chr “Boomerang (1992)” “Net, The (1995)” “Outbreak (1995)” “Stargate (1994)” ... $ genres : chr “Comedy|Romance” “Action|Crime|Thriller” “Action|Drama|Sci-Fi|Thriller” “Action|Adventure|Sci-Fi” ... - attr(*, “.internal.selfref”)=

```
# dimension of edx
dim(edx)
```

[1] 9000055 6

```
head(edx)
```

```
##    userId movieId rating timestamp
## 1:      1     122      5 838985046
## 2:      1     185      5 838983525
## 3:      1     292      5 838983421
## 4:      1     316      5 838983392
## 5:      1     329      5 838983392
## 6:      1     355      5 838984474
##                           title
## 1:              Boomerang (1992)
## 2:               Net, The (1995)
## 3:               Outbreak (1995)
## 4:               Stargate (1994)
## 5: Star Trek: Generations (1994)
## 6:       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
```

edx has 9000055 rows and 6 columns. This 6 columns contains our outcome to predict (rating) and features or predictors that will be used to predict our outcome (userId, movieId, timestamp, genres).

```
# Number of movies in our data set
n_distinct(edx$movieId)
```

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

```
#Number of users that are rated movies in our dataset
n_distinct(edx$userId)
```

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

The edx dataset contains **10 677 movies and 69 878 users**.

3.2 Ratings behavior

```
# No movie rated to 0
sum(edx$rating == 0)
```

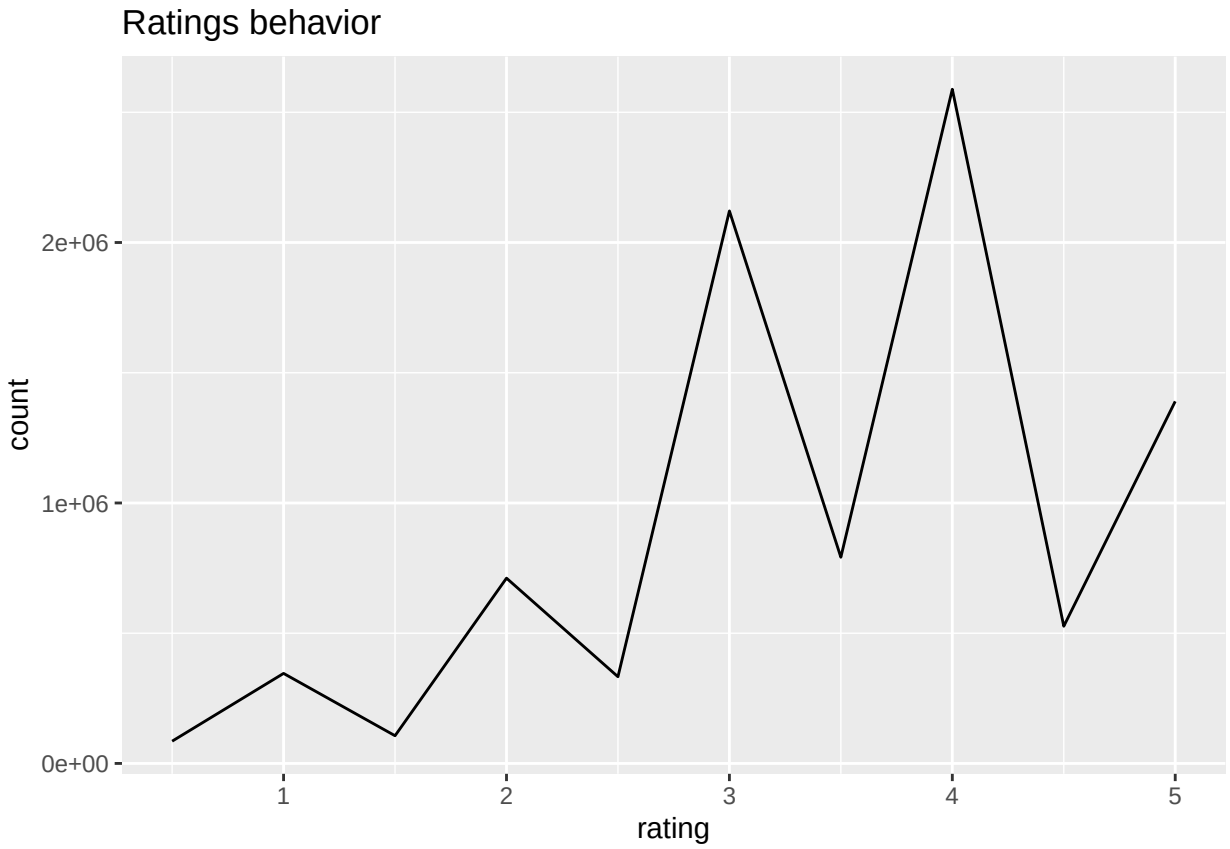
```
## [1] 0
```

```
#- Type of rating in movielens data set
n_distinct(edx$rating)
```

```
## [1] 10
```

All movies were rated at least to 0.5. 10 rating categories are possible (0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5). 3 and 4 rating are most popular for movies.

```
# Rating behavior in our data set

edx %>%
  group_by(rating) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = rating, y = count)) +
  geom_line()+
  ggtitle('Ratings behavior')
```

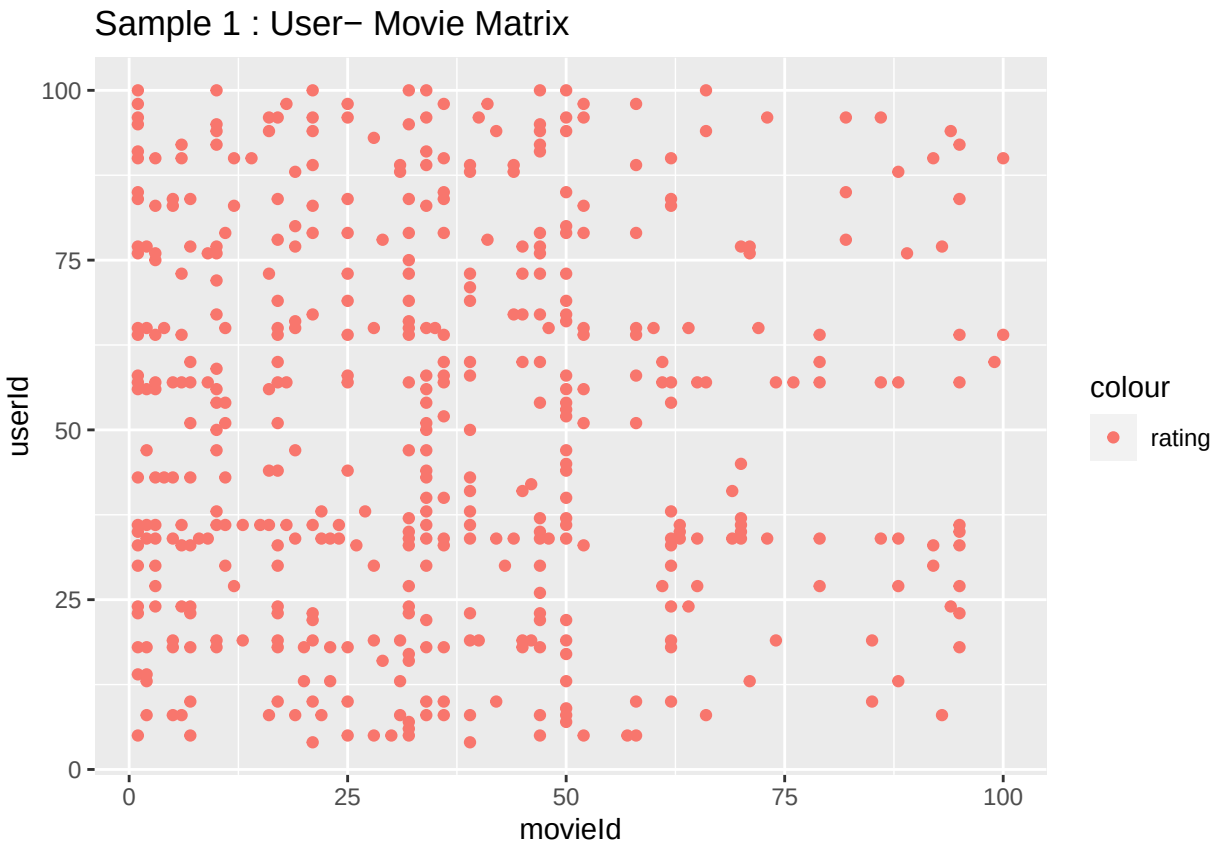


3.3 USERS AND MOVIES EFFECTS

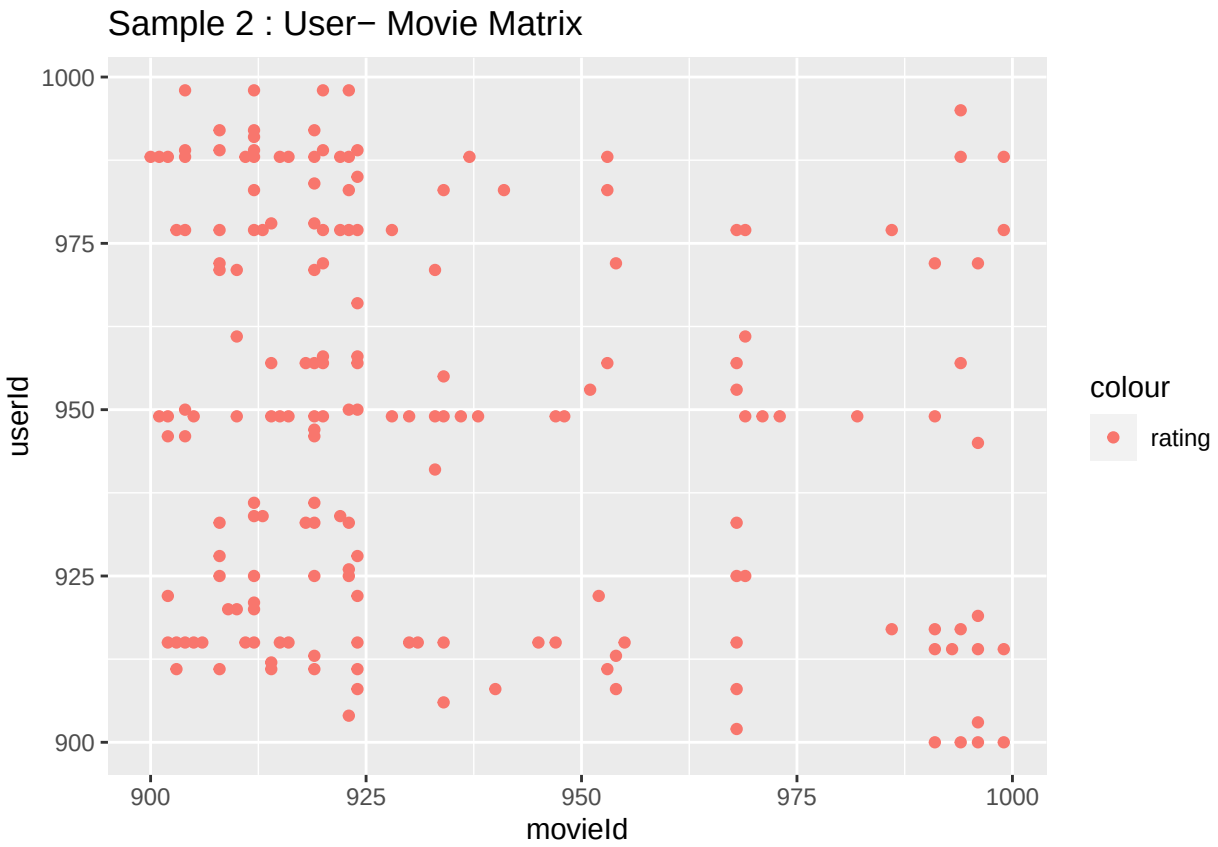
3.3 Users- Movies behavior

We have already now that we have much more users than movies. The figure below shows in two different samples that not all movies are rated by users. User- movie matrix is sparse with many empty cells.

```
# User- Movie matrix by rating for a sample of 100 users and movies with ID below 100.

edx%>% group_by(rating) %>% filter(userId<= 100 & movieId <= 100)%>%
  summarise(n=n(), movieId = movieId, userId =userId) %>%
  ggplot(aes(x=movieId, y= userId, color = 'rating'))+
  geom_point() +
  ggtitle('Sample 1 : User- Movie Matrix')
```

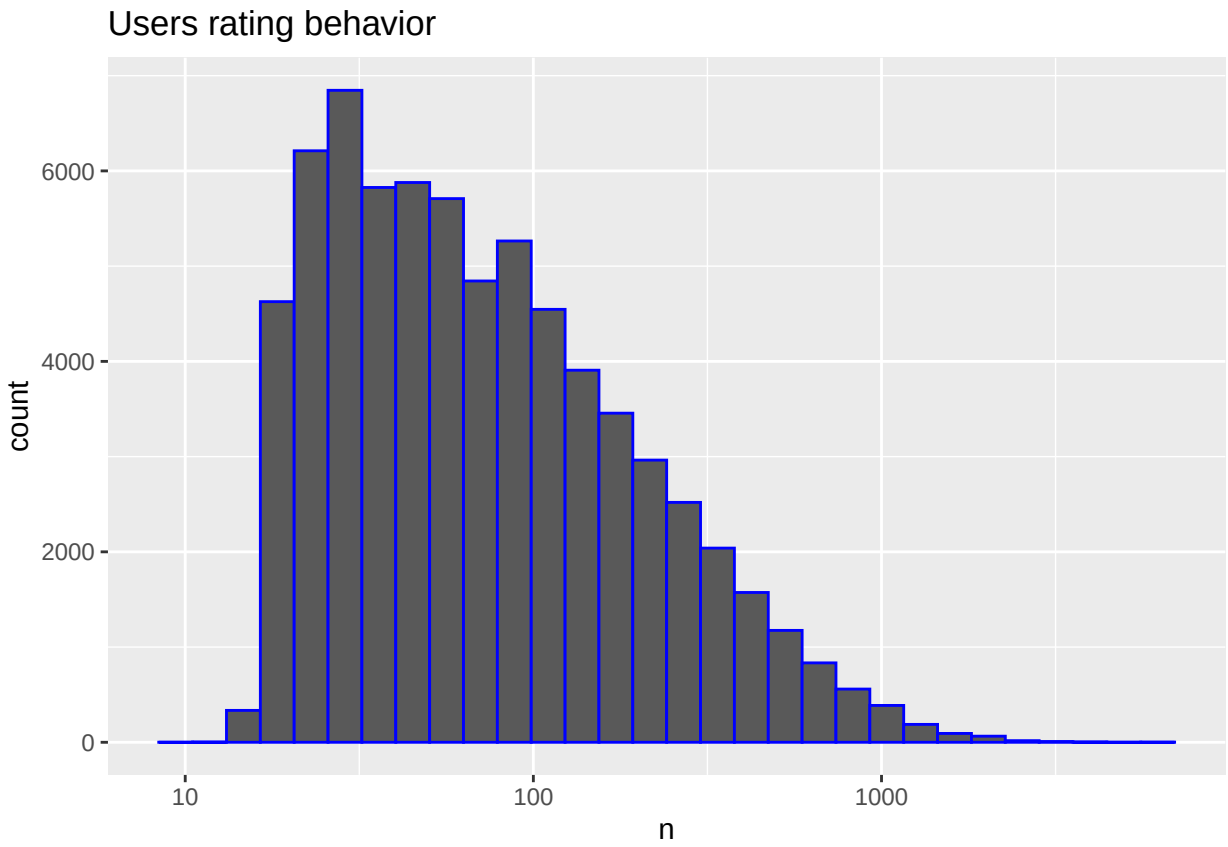



```
# User- Movie matrix by rating for a sample of 100 users and movies with ID between 900 and 1000.

edx%>% group_by(rating) %>%
  filter(userId<= 1000 & userId>= 900 & movieId <= 1000 & movieId>= 900)%>%
  summarise(n=n(), movieId = movieId, userId =userId) %>%
  ggplot(aes(x=movieId, y= userId, color = 'rating'))+
  geom_point()+
  ggtitle('Sample 2 : User- Movie Matrix')
```

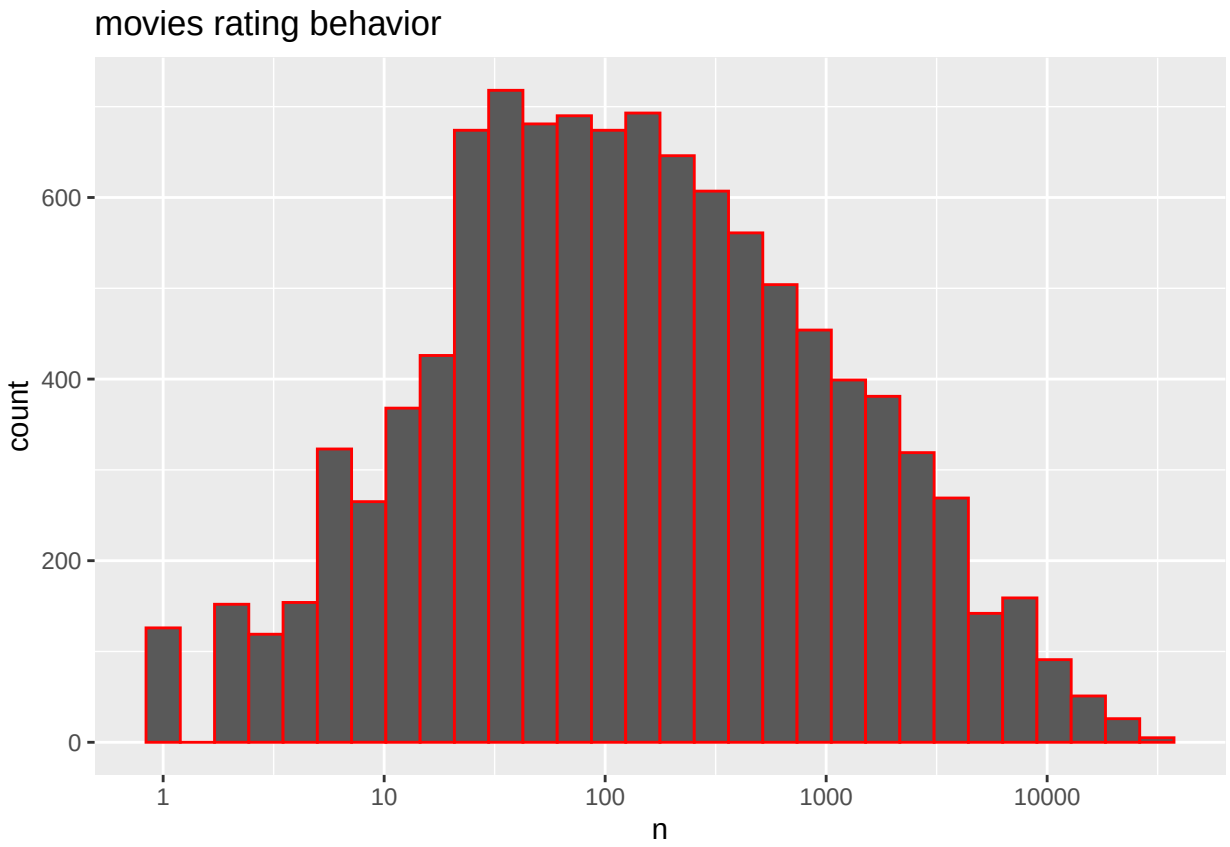


Some users are more active in ratings and some movies also get more ratings then others.

```
# Users rating behavior

edx %>% group_by(userId) %>% summarise(n =n()) %>%
  ggplot(aes(n))+
  geom_histogram(color='blue')+scale_x_log10()+ ggtitle('Users rating behavior')
```




```
# Movies rating behavior

edx %>% group_by(movieId) %>% summarise(n =n()) %>%
  ggplot(aes(n))+
  geom_histogram(color='red')+scale_x_log10()+ggtitle('movies rating behavior')
```



3.4 Movie's genres in our data set

3.4 Genres exploration

```
edx %>% group_by (genres) %>% summarise(n=n())
```

```
## # A tibble: 797 x 2
##    genres                                                n
##    <chr>                                             <int>
##  1 (no genres listed)                                    7
##  2 Action                                            24482
##  3 Action|Adventure                                  68688
##  4 Action|Adventure|Animation|Children|Comedy          7467
##  5 Action|Adventure|Animation|Children|Comedy|Fantasy   187
##  6 Action|Adventure|Animation|Children|Comedy|IMAX       66
##  7 Action|Adventure|Animation|Children|Comedy|Sci-Fi    600
##  8 Action|Adventure|Animation|Children|Fantasy          737
##  9 Action|Adventure|Animation|Children|Sci-Fi            50
## 10 Action|Adventure|Animation|Comedy|Drama             1902
## # ... with 787 more rows
```

For genres group, we have observe that some categories are much more rated than others. Another specificity of our genres features is that several genres are ,combined for each movie. We have 797 different combinations of genres.

```
# Arrange genres by number of ratings

edx %>% group_by (genres) %>% summarise(n=n())%>% arrange(desc(n))
```

```
## # A tibble: 797 x 2
##    genres                          n
##    <chr>                       <int>
##  1 Drama                      733296
##  2 Comedy                     700889
##  3 Comedy|Romance             365468
##  4 Comedy|Drama               323637
##  5 Comedy|Drama|Romance       261425
##  6 Drama|Romance              259355
##  7 Action|Adventure|Sci-Fi    219938
##  8 Action|Adventure|Thriller 149091
##  9 Drama|Thriller             145373
## 10 Crime|Drama                137387
## # ... with 787 more rows
```

```
edx %>% group_by (genres) %>% summarise(n=n())%>% arrange(n)
```

```
## # A tibble: 797 x 2
##    genres                                          n
##    <chr>                                       <int>
##  1 Action|Animation|Comedy|Horror                  2
##  2 Action|War|Western                              2
##  3 Adventure|Fantasy|Film-Noir|Mystery|Sci-Fi      2
##  4 Adventure|Mystery                               2
##  5 Crime|Drama|Horror|Sci-Fi                       2
##  6 Documentary|Romance                             2
##  7 Drama|Horror|Mystery|Sci-Fi|Thriller            2
##  8 Fantasy|Mystery|Sci-Fi|War                      2
##  9 Action|Adventure|Animation|Comedy|Sci-Fi        3
## 10 Horror|War|Western                             3
## # ... with 787 more rows
```

```
# Examples of Drama that is highly rated

drama_rating <-edx %>% group_by (genres) %>% summarise(n=n()) %>% filter(grepl('Drama', genres))
sum(drama_rating$n)
```

```
## [1] 3910127
```

This top ten combinations of genres shows that Drama, Comedy and Romance have many rating. For example, we can count **3910127 of rating for Drama genres**.

4. Modeling approach and global effects

edx data will be divide into a training set and a test set. Our different models will be tested in this partition for evaluating several improvements of the models by the calculations of RMSE.

The method will be linear regression models as machine learning algorithms.

In our movieLens data set, our outcome is the movie "rating". We have userId, movieId, genres, Timestamp that can be used as features or predictors. Are they effects in our rating outcome? We will verify this before using them in our models.

In the linear regression course, we have learned that adding extra predictors can improve RMSE substantially, but not when the added predictors are highly correlated with other predictors. We also know that reducing RMSE improves our linear regression model.

I will be then compare RMSE of several linear regression by adding predictors that have effects in ratings to achieve the RMSE goal.

4.1 Equations and models

Our models will be mainly based on linear regression models. They are popular in machine learning. We have relationship between our rating outcome and movie-user predictors.

We will use the loss function based on the Root Mean Squared Error (RMSE) as in the Netflix challenge to evaluate our models. Minimizing this RMSE error permits us to evaluate how our predicted ratings are closest to actual rating.

RMSE has the same unit than the outcome and is defined by the following equation with N, number of user/movie combination.$\hat{y}_{u,i}$ is the movie/user predicted rating and $y_{u,i}$ the actual rating for movie i by user u.

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

We have already demonstrate that movies and users have a strong effect on rating and that adding predictors can improve the RMSE. We will follow this arguments for models :

-Model 1 : naive approach

A first model is to assume the same rating for all movies and users with all the differences in rating explained by random variation.

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

-Model 2 : include the movie effect

As the strong effects of movies in rating already demonstrated in data exploration part, we should take account this information by adding a new term in our linear model : b_i. b_i is the effect of movie i in rating outcome. The notation b, refers to bias as in the Netflix challenge papers. We will use least squares to estimate b_i knowing that $\hat{b}_i$ is the average of $(Y_{u,i} - \hat{\mu})$ for movie i. For simplifying, the hat notation will not be used in this report.

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

-Model 3 : include the users effect

In the previous section, it is clear that users have non negligible effects on rating. We will add this users effects to our linear models. This models is aim to be more robust as its is a combination of the users and movies effects.

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

-Model 4 : regularize our movie and user linear model and choose the better tuning parameter lambda

Regularization will be applied to prevent over-training in our data. This will help us to constrain the total variability of the size effects. Small sizes of users rating can induce a very small bias of movie effect. Some users are rarely rating movies. Some movies can have just one user rating.

We will penalized this small sizes by adding penalty terms to our movie-user model and then control the total variability of movies and users effects. Penalty terms to be add to equation (4) are $\lambda \sum_i {b_i}^2$ and $\lambda \sum_u {b_u}^2$.

Regularization will be made in two steps in our user_movie effect. A penalty term is first add to our movie effect bias and then to our movie_users effect bias.

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

$$\hat{b_u}(\lambda) = \frac{1}{\lambda + n_u} \sum_{i=1}^{n_u} (Y_{u,i} - \hat{b_i}(\lambda) - \hat{\mu}) \tag{6}$$

Lambda is a tuning parameter. Cross validation can be used for choosing the lambda that minimizes RMSE. This optimum lambda will be used to calculated the RMSE of regularized movie/user effects.

This Model 4 will be trained only in edx dataset.It will result in an optimized RMSE with the best lambda.

If the RMSE result of this model 4 is close to our goal. Ie will go to the final validation. As overfitting is already take account in my regularized model. The final RMSE should be improved when it is calculated in validation set.

-Final Model : Movie- user linear model regularized with best lambda with validation set

My final model for this report will be based on model 4 as a linear regression model regularized with lambda that given minimized RMSE.

The validation set can be now used to provide our final RMSE.

4.2 Split edx into training and test set

Validation set is considered as our new dataset where the outcome is unknow.

For testing our models, we will use only edx data set where rating is know. It is usually the case in machine learning model construction. We will split the edx data set randomly in edx_training and edx_test using the following codes. edx_test will be 20% of edx data.

```
##Split edx data into training and 20% test set


set.seed(1, sample.kind = 'Rounding')
edx_test_index <- createDataPartition(y= edx$rating, times = 1, p= 0.2, list = FALSE)
edx_training <- edx[- edx_test_index, ]
temp_edx <- edx[edx_test_index, ]

#Make sure userId and movieId in test set are also in training set

edx_test <- temp_edx  %>% semi_join(edx_training, by = "movieId") %>%  semi_join(edx_training, by = "use

# Add rows removed from test set back into training set

 removed_edx <- anti_join(temp_edx, edx_test)
 edx_training <- rbind(edx_training, removed_edx)
rm( edx_test_index, temp_edx, removed_edx)
```

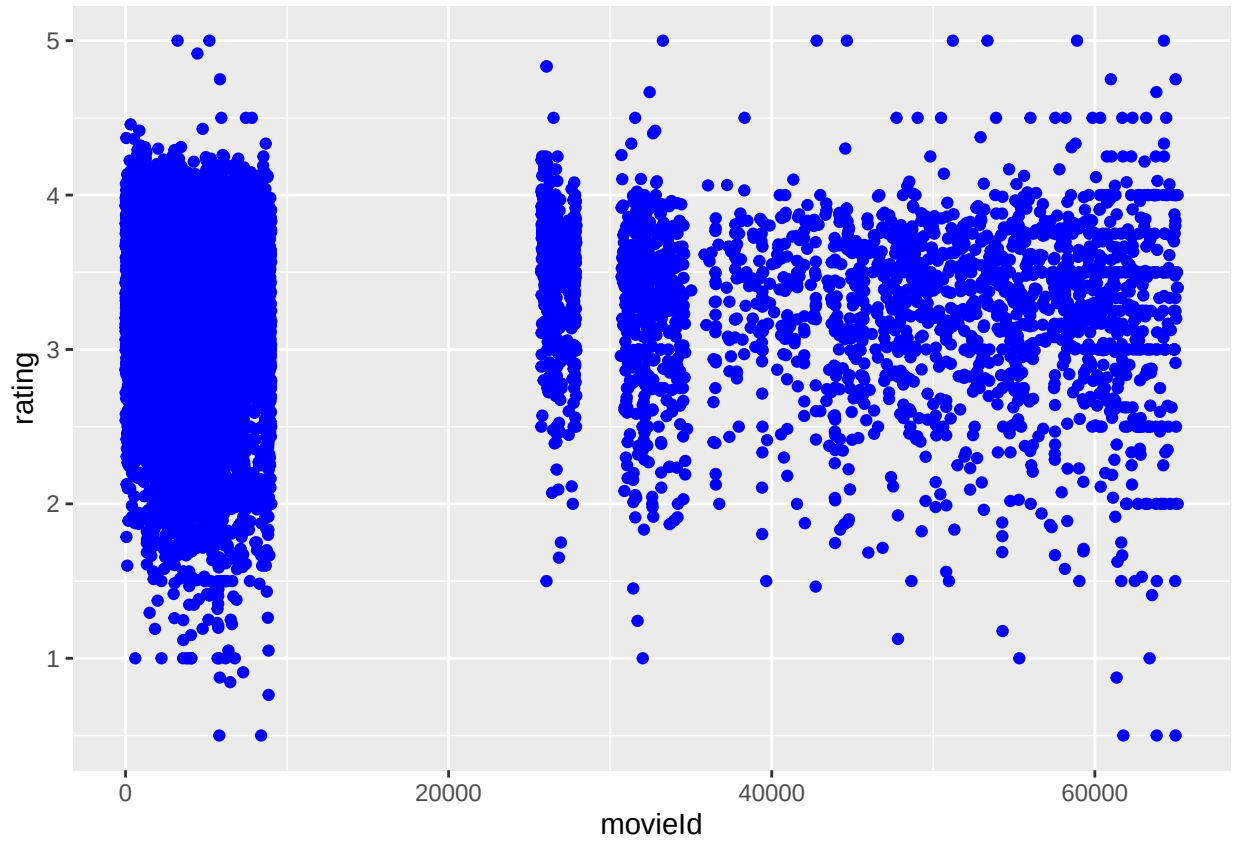
4.3 Exploration of Global effects in movie's rating

In this chapter, we will plot and analyze the predictors effects on our movie rating on the edx training data. Let's compute the average of edx_training rating.

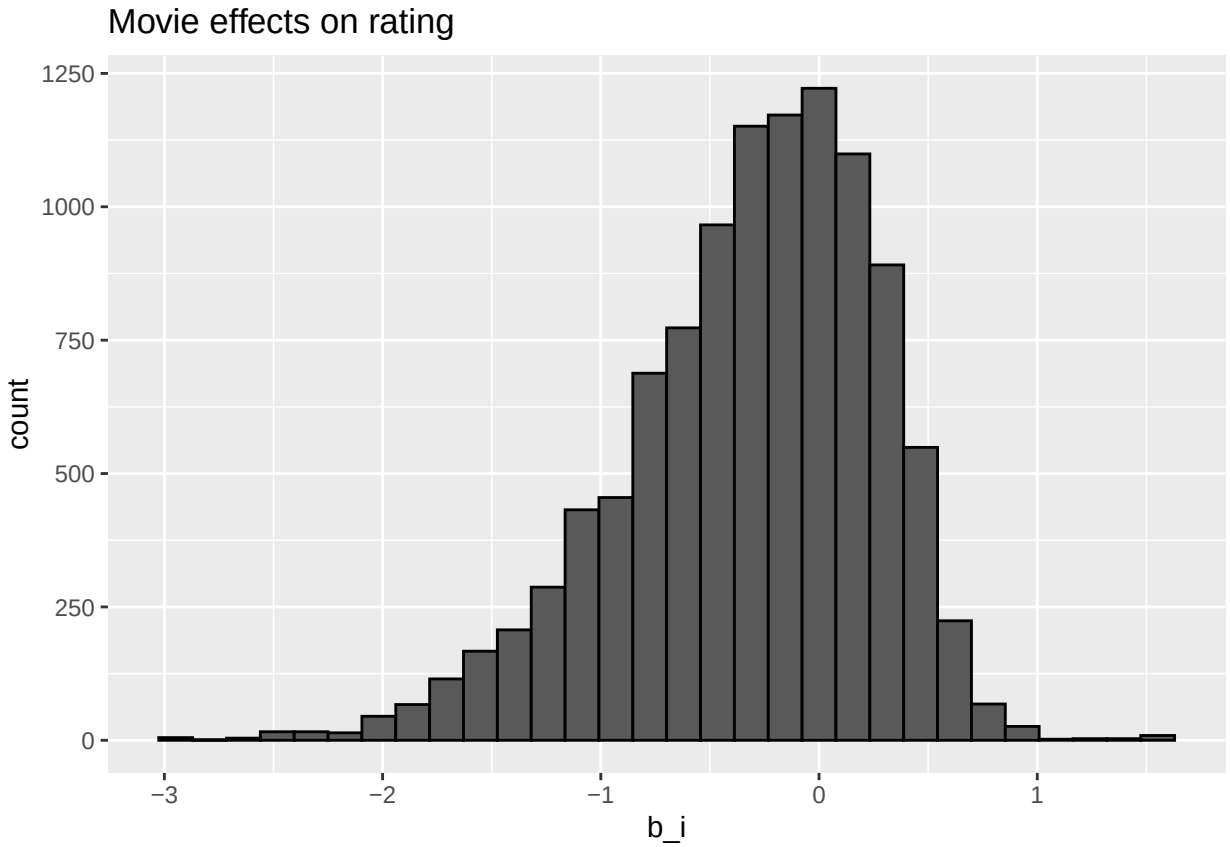
```
# Average of ratings in edx training data
mu_hat <- mean(edx_training$rating)
```

4.3.1 Movies effects on rating

```
## b_i is the effect of each movie on rating.
##We know that some movies are most popular than others and are able to get more ratings/ higher rating

movie_effect <- edx_training %>% group_by(movieId)%>%
  summarize(b_i = mean(rating- mu_hat), rating = mean(rating))

movie_effect %>% ggplot(aes(movieId, rating))+ geom_point( color = 'blue')
```



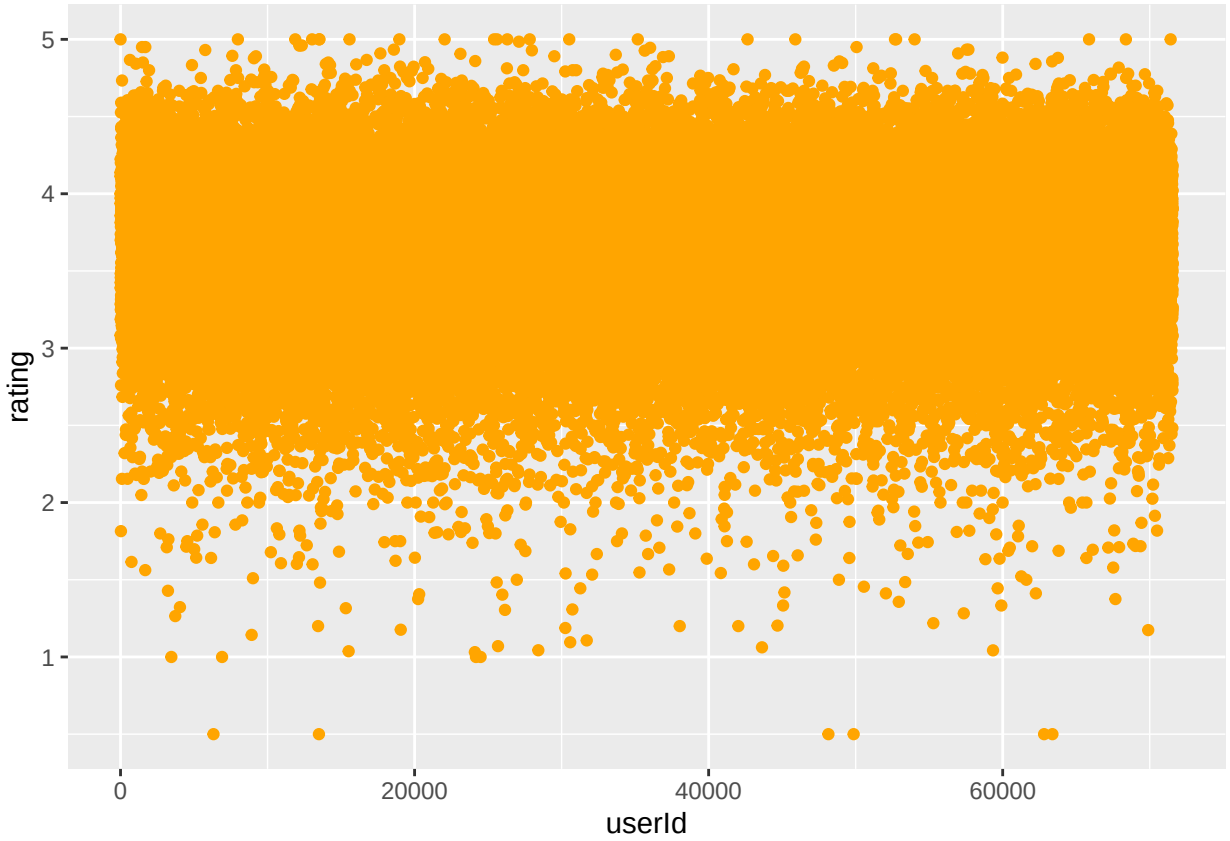

```
##Plot the bias b_i for movie i
movie_effect %>% ggplot(aes(b_i)) +
  geom_histogram(bins = 30, color = "black") +ggtitle('Movie effects on rating')
```



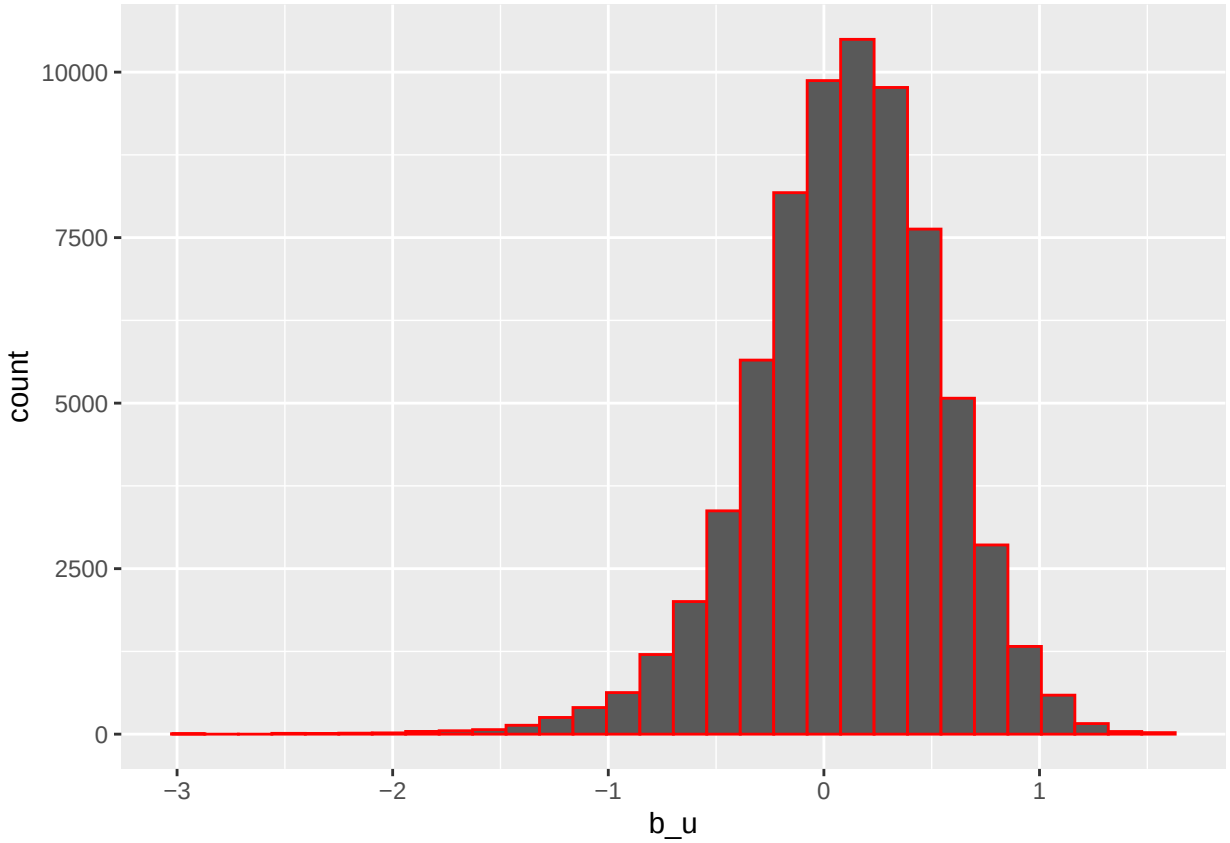
This figure shows a substantial distribution of movie effect on rating. Some movies are pretty awesome and are rated higher than normally expected. The movieId/ rating plot demonstrates very well that some movies are much more rated then others.

4.3.2 Users effects on rating

```
## Users effects on rating
## b_u is the effect of each user on rating.
##We know that some users tend to rate movies lower than the average user.

user_effect <- edx_training %>% group_by(userId)%>%
  summarize( b_u = mean(rating- mu_hat),rating = mean(rating))

user_effect %>% ggplot(aes(userId, rating)) + geom_point(color ='orange')
```



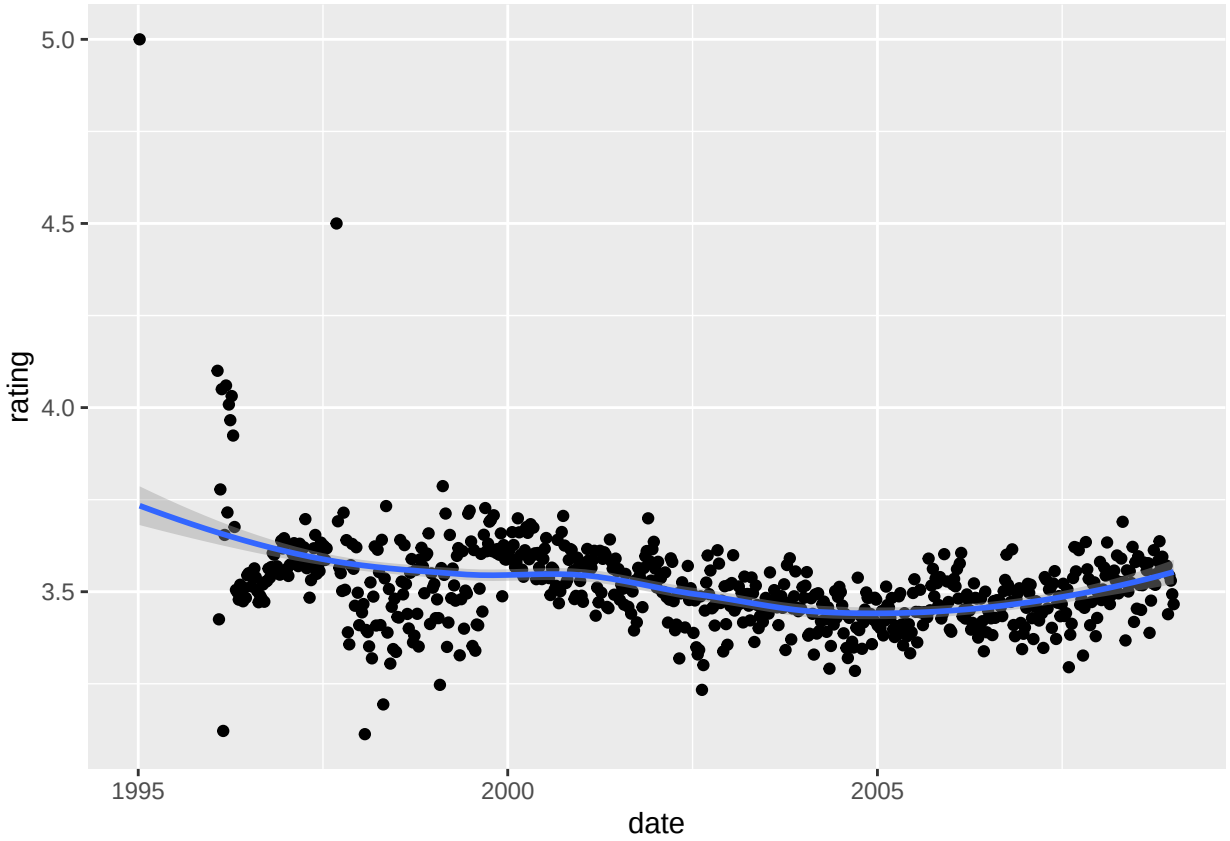

```
# Plot the user specific effect b_u
user_effect %>% ggplot(aes(b_u))+ geom_histogram(bins = 30, color = "red")
```



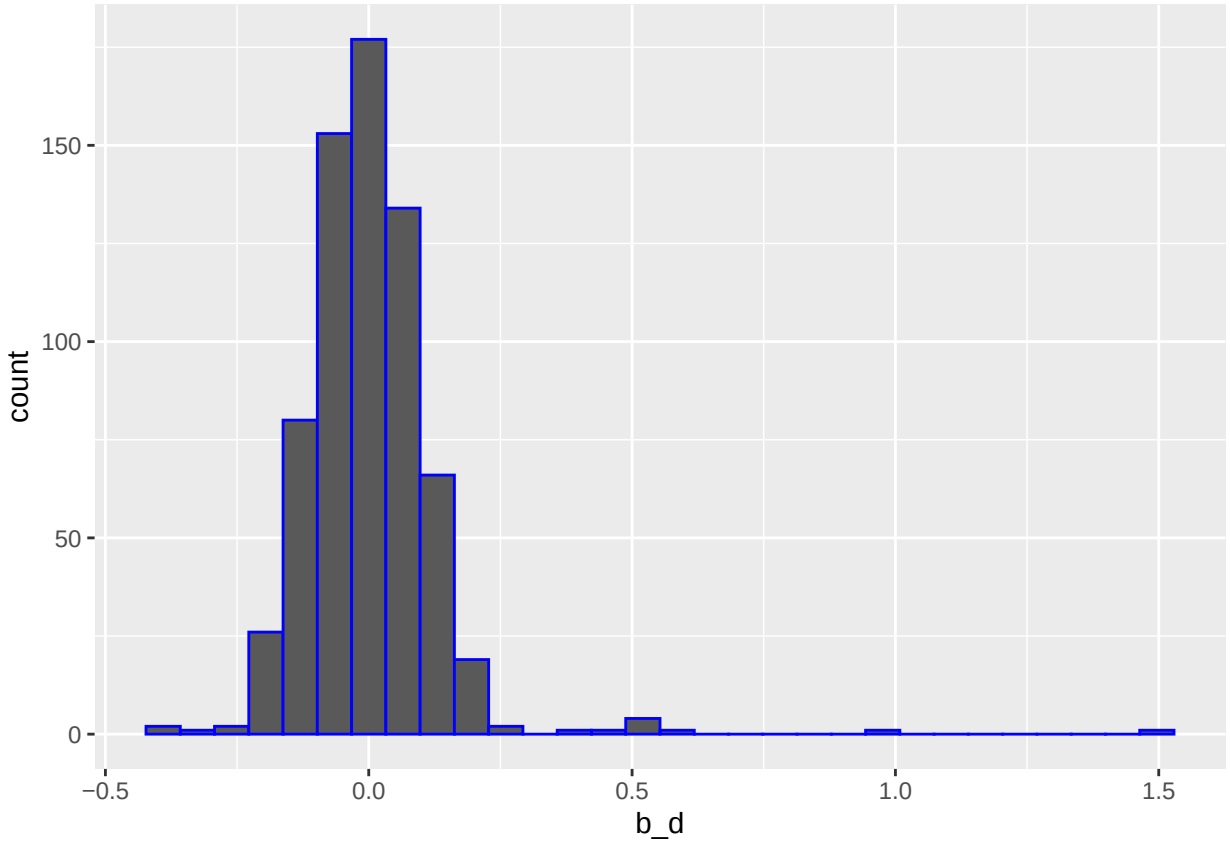
Ratings is highly correlated to users. Some users are much more engage in rating then others and are rating lower or higher than the normally expected. A strong effect in rating by users should be considered.

4.3.3 Dates effects on rating

```
##  Date effects on rating

# Install lubridate packages
library(lubridate)

#Transform our timestamp as a datetime
edx_training <- mutate( edx_training, date = as_datetime(timestamp))

date_effect <- edx_training %>% mutate(date = round_date(date, unit = "week"))%>%
  group_by(date) %>% summarize(b_d = mean(rating- mu_hat),rating = mean(rating))

date_effect %>% ggplot(aes(date, rating)) + geom_point() + geom_smooth()
```



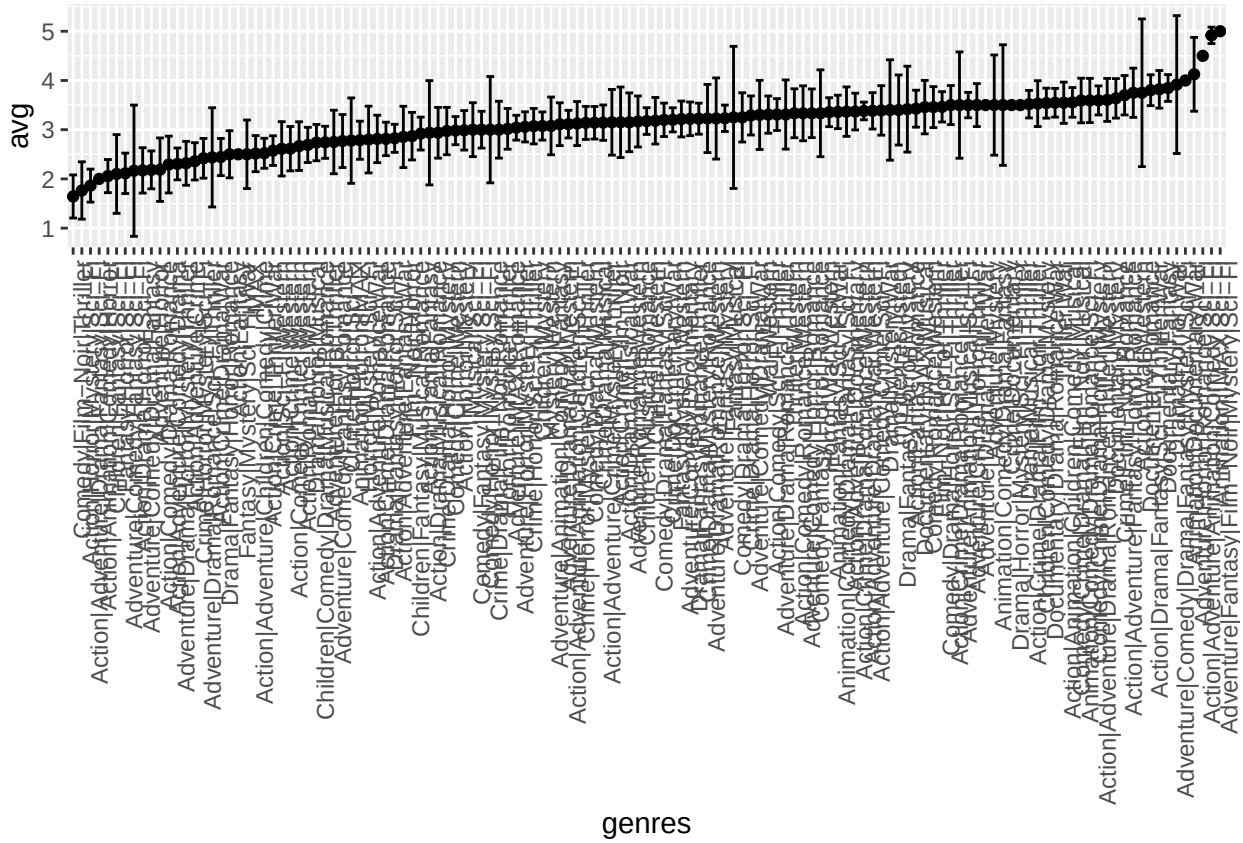

```
# Plot date specific effect on rating b_d
date_effect %>% ggplot(aes(b_d)) + geom_histogram(bins = 30, color = "Blue")
```



edx data set contains rating from **1995**. The date has effect in rating but not a strong effect.

4.3.4 Genres effects on rating

```
#Genres effects on rating less then 150 movies

edx_training %>% group_by(genres) %>%
  summarize(n = n(), avg = mean(rating), se = sd(rating)/sqrt(n())) %>%
  filter(n < 50) %>%
  mutate(genres = reorder(genres, avg)) %>%
  ggplot(aes(x = genres, y = avg, ymin = avg - 2*se, ymax = avg + 2*se)) +
  geom_point() +
  geom_errorbar() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Genres group or combinations have a strong effect in rating. Some genres have huge variation in their ratings.

5.Training models and results

5.1 Model 1- Naive prediction

Let's train our first naive model that supposes a same rating for all movies and users. We will compute rmse() function directly with the metrics library.

```
library(Metrics)
mu_hat <- mean(edx_training$rating)
naive_rmse <- rmse(edx_test$rating, mu_hat)
```

For a easier following of our RMSEs results,let's create a results table.

```
## # A tibble: 1 x 2
##   method                     RMSE
##   <chr>                     <dbl>
## 1 Model 1- Just the average  1.06
```

The RMSE of model 1 is around 1 and is so far to our goal : RMSE< 0.8649.

Let's see which improvement will occur if I add a movie effect to the first linear model.

5.2 Model 2- Include the movie effect

```
#Compute the movie effect b_i
movie_effect <- edx_training %>% group_by(movieId)%>%
```

```
  summarize(b_i = mean(rating- mu_hat))

# prediction of rating considering movie effect
pred_movie <- mu_hat + edx_test %>% left_join(movie_effect, by='movieId') %>% .$b_i

#RMSE of the model 2 taking account the movie effect
movie_rmse <- rmse(edx_test$rating, pred_movie)
```

```
## # A tibble: 2 x 2
##   method                      RMSE
##   <chr>                      <dbl>
## 1 Model 1- Just the average 1.06
## 2 Model 2 - Movie effect    0.944
```

Our model is slighty improved with an RMSE of 0.9437429

5.3 Model 3- Add the users effect to movie effect

```
# As we now that adding extra predictors- can improve our RMSE
# We will add the user effects in our Model

#Let's first report our movie effect
movie_effect <- edx_training %>% group_by(movieId)%>%
  summarize(b_i = mean(rating- mu_hat))

# Include now user's effect with movie effect
user_effect <- edx_training %>% left_join(movie_effect, by = "movieId") %>%
  group_by(userId)%>% summarize( b_u = mean(rating- b_i- mu_hat))

# prediction of rating considering movie and user effects
pred_movie_user <- edx_test %>% left_join(movie_effect, by='movieId') %>%
  left_join(user_effect, by = 'userId')%>% mutate(pred = mu_hat +b_i + b_u)%>%
  pull(pred)

#RMSE of the model 3 taking account the movie and user effects
movie_user_rmse <- rmse(pred_movie_user, edx_test$rating)
```

```
## # A tibble: 3 x 2
##   method                         RMSE
##   <chr>                         <dbl>
## 1 Model 1- Just the average     1.06
## 2 Model 2 - Movie effect        0.944
## 3 Model 3 - Movie+ User effects 0.866
```

We have substantially improve our RMSE to 0.8659319 'and start to be closer to our goal of 0.8649. This RMSE result fit with our assumptions of adding predictors to our linear model is a good way of improvements.

5.4 Model 4- regularization of movies- users effects

-choose the better tuning parameter lambda

Which best lamdba will permit us to regularized the sample sizes effect. A cross validation with a lambda between 0 and 10 is compute in the following code.

```
lambdas <- seq(0,10, 0.25)
```

```
## Calculate RMSEs of all values of lambda for the regularized user/movie effects
lambda_rmses <- sapply(lambdas, function(l) {
  mu_hat <- mean(edx_training$rating)

  #Regularized movies effect
  movie_effect_r <- edx_training %>% group_by(movieId)%>%
    summarize(b_i_r = sum(rating- mu_hat)/(n()+l))

  #Regularized users effect
  user_effect_r <- edx_training %>% left_join(movie_effect_r, by= 'movieId')%>%
    group_by(userId)%>% summarize( b_u_r = sum(rating- b_i_r - mu_hat)/(n()+l))

  # prediction of rating considering regularized movie and user effects
  pred_movie_user_reg <- mu_hat + (edx_test %>%
                                     left_join(movie_effect_r, by='movieId') %>%
                                     left_join(user_effect_r, by = 'userId')%>%
                                     mutate(bi_u_r =b_i_r + b_u_r)%>%
                                     pull(bi_u_r))
   #Compute RMSEs for each lambda
  return( rmse(pred_movie_user_reg, edx_test$rating) )
})

# Compute lambda that minimizes the RMSE of regularized movie-user effects
lambda <- lambdas[which.min(lambda_rmses)]
lambda
```

```
## [1] 4.75
```

When applying lambda to our movie-users effects, we obtain a slightly improved RMSE regarding only training the non-regularized movie-users combinations.

We have now an optimized value of λ= 4.75. We can train our Model 5 that will be used at the final validation set.

-Movie- user linear model regularized with best lambda only with edx dataset

```
# Movie effect regularized with lambda
movie_effect_l <- edx_training %>% group_by(movieId)%>%
  summarize(b_i_l = sum(rating- mu_hat)/(n()+lambda))

#User and movie effects regularized with lambda
user_effect_l <- edx_training %>% left_join(movie_effect_l, by= 'movieId')%>%
  group_by(userId)%>% summarize( b_u_l = sum(rating- b_i_l - mu_hat)/(n()+lambda))

# prediction of rating considering regularized movie and user effects
pred_movie_user_l <- mu_hat + (edx_test %>% left_join(movie_effect_l, by='movieId') %>%
                                 left_join(user_effect_l, by = 'userId')%>%
                                 mutate(bi_u_l =b_i_l + b_u_l)%>%
                                 pull(bi_u_l))

#RMSE of the model 5 taking account the best lambda of regularized movie and user effects
movie_user_lambda_rmse <- rmse(pred_movie_user_l, edx_test$rating)
```

```
## # A tibble: 4 x 2
##   method                                  RMSE
##   <chr>                                  <dbl>
```

```
## 1 Model 1- Just the average                  1.06
## 2 Model 2 - Movie effect                     0.944
## 3 Model 3 - Movie+ User effects              0.866
## 4 Model 4 - Regularized Movie+ User effects 0.865
```

5.5 Final Model- movie and user linear model regularized with the best lambda on validation set

```
# mu correspond to the average rating on all edx data
mu <- mean(edx$rating)

# The final movie effect calculated on edx  and regularized using lambda
movie_effect_final <- edx %>% group_by(movieId) %>%
  summarize(b_i_f = sum(rating- mu)/(n()+ lambda))

# The final user and movie effects calculated on edx  and regularized using lambda
user_effect_final <- edx %>% left_join(movie_effect_final, by= 'movieId')%>%
  group_by(userId)%>% summarize( b_u_f = sum(rating- b_i_f - mu)/(n()+lambda))

# Final prediction of rating considering regularized movie and user effects
prediction_rating <- mu + (validation %>% left_join(movie_effect_final, by='movieId') %>%
                             left_join(user_effect_final, by = 'userId')%>%
                             mutate(bi_u_f =b_i_f + b_u_f)%>%
                             pull(bi_u_f))

# compute final RMSE of my project
Final_rmse <- rmse(prediction_rating, validation$rating)
```

```
## # A tibble: 6 x 2
##   method                                      RMSE
##   <chr>                                      <dbl>
## 1 Model 1- Just the average                  1.06
## 2 Model 2 - Movie effect                     0.944
## 3 Model 3 - Movie+ User effects              0.866
## 4 Model 4 - Regularized Movie+ User effects  0.865
## 5 Final Model - Regularized Movie+ User effects 0.865
## 6 PROJECT GOAL                               0.865
```

5.6 Tests models

```
## Test and highlight the best 10 movies

validation%>% left_join(movie_effect_final, by='movieId') %>%
  left_join(user_effect_final, by = 'userId')%>% mutate(pred =mu+ b_i_f + b_u_f)%>%
  arrange(-pred)%>% group_by(title)%>% select(title, genres)%>% head(10)
```

```
## # A tibble: 10 x 2
## # Groups:   title [7]
##    title                                        genres
##    <chr>                                        <chr>
##  1 Usual Suspects, The (1995)                   Crime~
##  2 Shawshank Redemption, The (1994)             Drama
##  3 Shawshank Redemption, The (1994)             Drama
##  4 Shawshank Redemption, The (1994)             Drama
```

```
##  5 Eternal Sunshine of the Spotless Mind (2004)       Comed~
##  6 Star Wars: Episode IV - A New Hope (a.k.a. Star W~ Actio~
##  7 Schindler's List (1993)                            Drama~
##  8 Donnie Darko (2001)                                Drama~
##  9 Star Wars: Episode VI - Return of the Jedi (1983)  Actio~
## 10 Schindler's List (1993)                            Drama~
```

```
## Highlight the 10 worst rated movies

validation%>% left_join(movie_effect_final, by='movieId') %>%
  left_join(user_effect_final, by = 'userId')%>% mutate(pred =mu+ b_i_f + b_u_f)%>%
  arrange(desc(-pred))%>% group_by(title)%>% select(title, genres)%>% head(10)
```

```
## # A tibble: 10 x 2
## # Groups:   title [9]
##    title                                          genres
##    <chr>                                          <chr>
##  1 Battlefield Earth (2000)                       Action|Sci-Fi
##  2 Police Academy 4: Citizens on Patrol (1987)    Comedy
##  3 Karate Kid Part III, The (1989)                Action|Adven~
##  4 Pokémon Heroes (2003)                          Animation|Ch~
##  5 Turbo: A Power Rangers Movie (1997)            Action|Adven~
##  6 Kazaam (1996)                                  Children|Com~
##  7 Pokémon Heroes (2003)                          Animation|Ch~
##  8 Shanghai Surprise (1986)                       Adventure
##  9 Free Willy 3: The Rescue (1997)                Adventure|Ch~
## 10 Steel (1997)                                   Action
```

6. Conclusion and recommendations

This is my first report with Rmarkdown, a very powerful tool that I have explore through this capstone. The project allow me to practice and confirm many skills in data science.

Recommendations systems are very common nowadays. This skills can be easily expand to a huge field when recommending products to consumers make sense.

With simplest linear regression model, we can achieve a good RMSE(Root Mean Square Error) of 0.8648201. It means that we are able to propose a machine learning algorithm that minimizes very well the error between the prediction and actual rating based on users and movies effects.

For next steps, it will be interesting for me to train others machine learning algorithms as Matrix factorization. In my report, I have take account the movies differences effects with b_i and the users differences effects with b_u. I have not take account on the variations related to similar rating by groups of movies and group of users. The matrix Factorization is widely used on recommendation systems and was used on the Netflix challenge.

Limitations

The first part of data exploration was helpful as a guidance of predictors effects on the outcome and how to include predictors for models improvements. I have work only with two predictors. The analysis show that we have a strong genres effects on ratings. Including genres effects on the linear model will surely improve it. The model was not fit with lm() function that would be very slow.

To go further, many software specifically design for recommendations systems algorithms are available to explore. An example is the recommenderlab package focus on collaborative filtering.