Predicting Movie Ratings in the "MovieLens Dataset"

project written by Samuel Beqiri

Contents

Introduction.....2

Project objective.....2

Data set overview.....3

Summary Table.....4

Ratings.....5

Movies.....6

Users.....10

Genres.....11

Model analysis "RMSE".....14

Results.....16

Final Validation.....17

Conclusions.....18

Introduction

The “MovieLens Dataset” is a stable benchmark dataset provided by the GroupLens research lab in the Department of Computer Science and Engineering at the University of Minnesota, Twin Cities. The GroupLens lab specializes in recommender systems, online communities, mobile and ubiquitous technologies, digital libraries, and local geographic information systems. Users were selected randomly for inclusion in the dataset, with ratings taken from the full set of MovieLens data. All users selected had rated at least 20 movies. No user demographic information is included - each user is represented by an id, and no other information is provided. The goal of this project will be to predict ratings for unseen data. A model will be built from the training data, and then applied to the unseen test data. The success metric will be root mean square estimate (RMSE). We are attempting to predict movie ratings on a scale of 0.5 stars to 5 stars, and RMSE will be measured on the same scale. An RMSE of 0 means we are always correct, which is unlikely. An RMSE of 1 means the predicted ratings are off by 1 star. The goal for this project is to achieve RMSE < 0.9849 as computed on the unseen test dataset.

Project objective

The objective for this project are as following:

- To create a movie recommendation system and reduce the RMSE less than 0.9849.
- To practice skills data wrangling, data visualization, reporting using tidyverse, ggplot2, rmarkdown.

Data set overview

The data set is generated by following code.

```
#Create edx set, validation set
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")
#MovieLens 10M dataset:
#https://grouplens.org/datasets/movielens/10m/
#http://files.grouplens.org/datasets/movielens/ml-10m.zip
dl <- tempfile()download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml 10M100K/ratings.dat"))),col.names = c("userId",
"movieId", "rating", "timestamp"))
movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")movies <- as.data.frame(movies) %>% mutate(movieId =
as.numeric(levels(movieId))[movieId],                                    title = as.character(title), genres = as.character(genres))
movielens <- left_join(ratings, movies, by = "movieId")
#Validation set will be 10% of MovieLens data
set.seed(1, sample.kind = "Rounding")
#if using R 3.5 or earlier, use `set.seed(1)` instead
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)
```

The edx file have 9000055 rows and 6 columns. There are 6 columns where:

- **userId** is an integer, identifies an user.
- **movieId** is an integer, identifies an movie.
- **title** is character, represents name of the movie.
- **rating** is an integer, ranging from 1 to 5, made on 5-stars scale (whole and half-star ratings).
- **timestamp** is represented in second since 1/1/1970 UTC.
- **genres** is character, represents the genre.

There are total 69,878 users, 10,677 movies, 797 genres, the data is recorded in 14 years from the first rating date 1995-01-09 to the last rating date 2009-01-05.

Summary Table

There are 2 datasets used in this project.

- The **edx** dataset is used to analyze the predictors, find insights and develop the recommendation system.
- The **validation** dataset is used to validate effectiveness of our developed recommendation system.

An overview on edx and validation datasets is provided as following.

```
#create edx summary table
edx_summary <- data.frame(number_of_rows = nrow(edx), number_of_column = ncol(edx), number_of_users = n_distinct(edx$userId), number_of_movies = n_distinct(edx$movieId), average_rating = round(mean(edx$rating),2), number_of_genres = n_distinct(edx$genres), the_first_rating_date = as.Date(as.POSIXct(min(edx$timestamp), origin = "1970-01-01")), the_last_rating_date = as.Date(as.POSIXct(max(edx$timestamp), origin = "1970-01-01")))edx_summary[,1:6] %>%  kable(caption = "Summary of edx set (part 1)") %>%  kable_styling(font_size = 10, position = "center", latex_options = c("scale_down","HOLD_position"))
```

number_of_rows	number_of_column	number_of_users	number_of_movies	average_rating	number_of_genres
9000055	6	69878	10677	3.51	797

```
#create validation summary table
validation_summary <- data.frame(number_of_rows = nrow(validation), number_of_column = ncol(validation), number_of_users = n_distinct(validation$userId),number_of_movies = n_distinct(validation$movieId), average_rating = mean(validation$rating),number_of_genres = n_distinct(validation$genres),the_first_rating_date = as.Date(as.POSIXct(min(validation$timestamp),  origin = "1970-01-01")),  the_last_rating_date = as.Date(as.POSIXct(max(validation$timestamp), origin = "1970-01-01")))
validation_summary[,1:6] %>% kable(caption = "Summary of validation set (part 1)", digits = 2) %>% kable_styling(font_size = 10, position = "center", latex_options = c("scale_down", "HOLD_position")
```

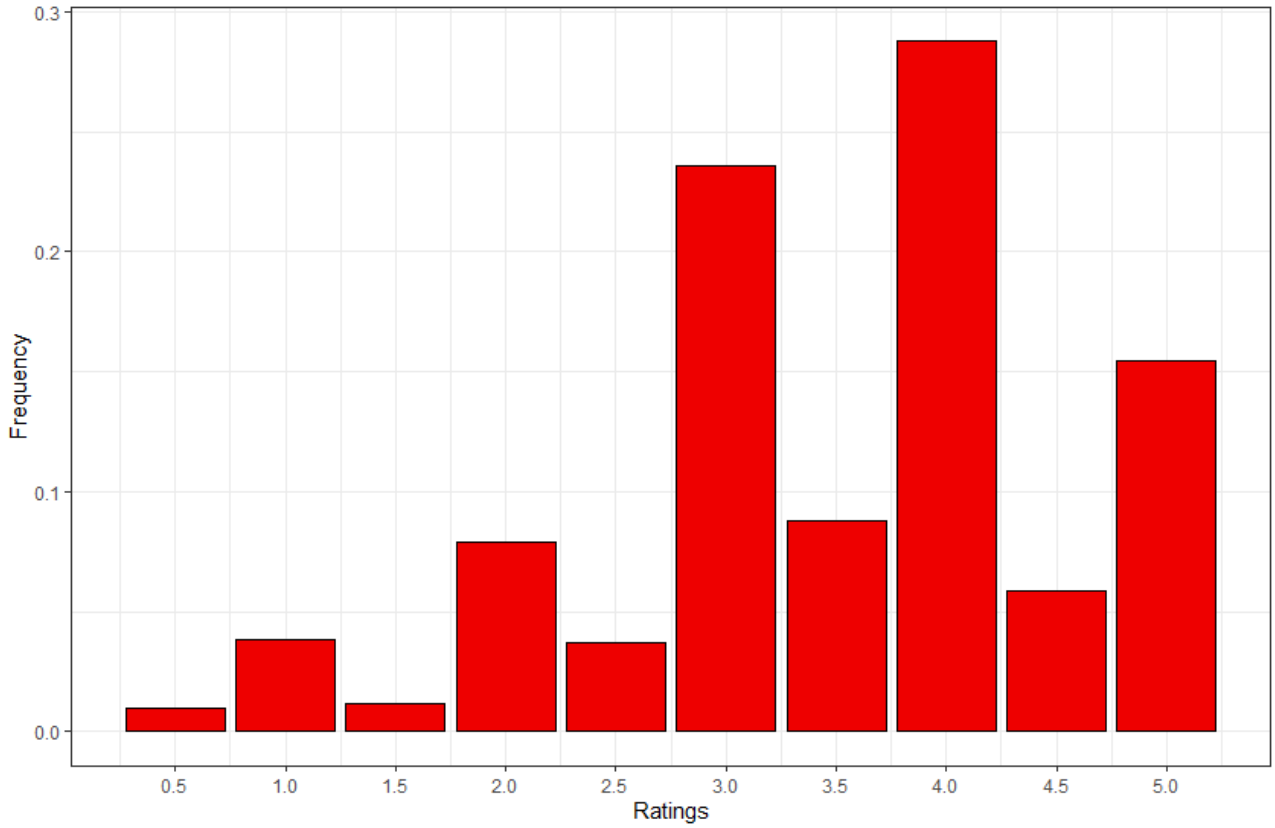
number_of_rows	number_of_column	number_of_users	number_of_movies	average_rating	number_of_genres
999999	6	68534	9809	3.51	773

Ratings

In our data set, rating is given by users for the movie they watched. The rating is an ordinal scale of number from 0.5 to 5. If they like the movie, they can give 5 stars or if they don't like the movie, they can give lower value.

```
#create a summary table grouping by rating
rating_sum <- edx %>% group_by(rating) %>% summarize(count = n())
gg <- rating_sum %>% mutate(rating = factor(rating)) %>% ggplot(aes(rating, count)) + geom_col(fill = "red2", color = "black") +  theme_classic() + labs(x = "Ratings", y = "Frequency", title = "Number of rating", caption = "Figure 1 - Rating in edx dataset")
ggplotly(gg)
```

Number of Ratings

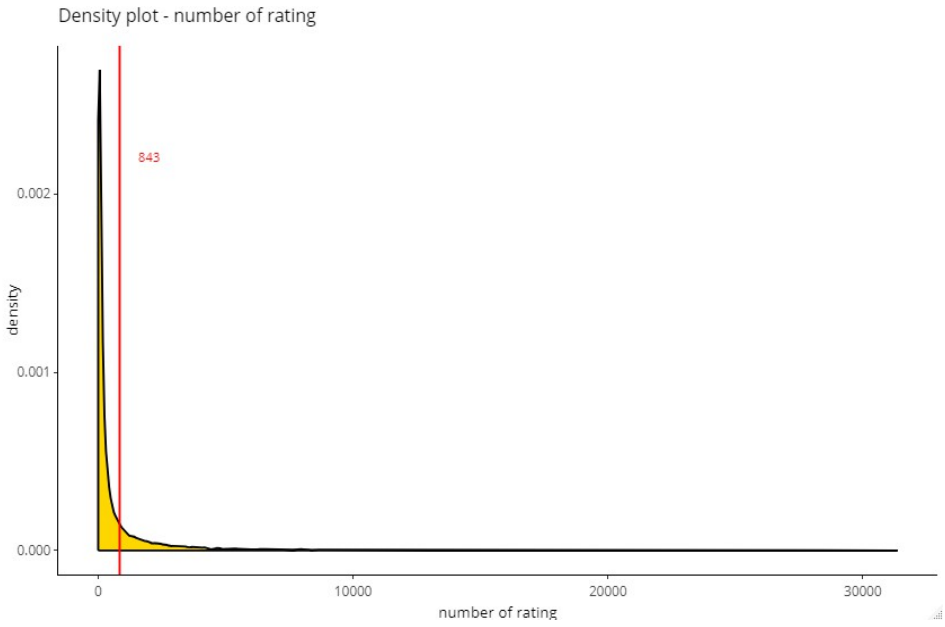


We can see in figure that in general, half star ratings are less common than whole star ratings. The most rating value given by users is 4.

Movies

The edx data set have total 10,677 movies, each is represented by a movieId

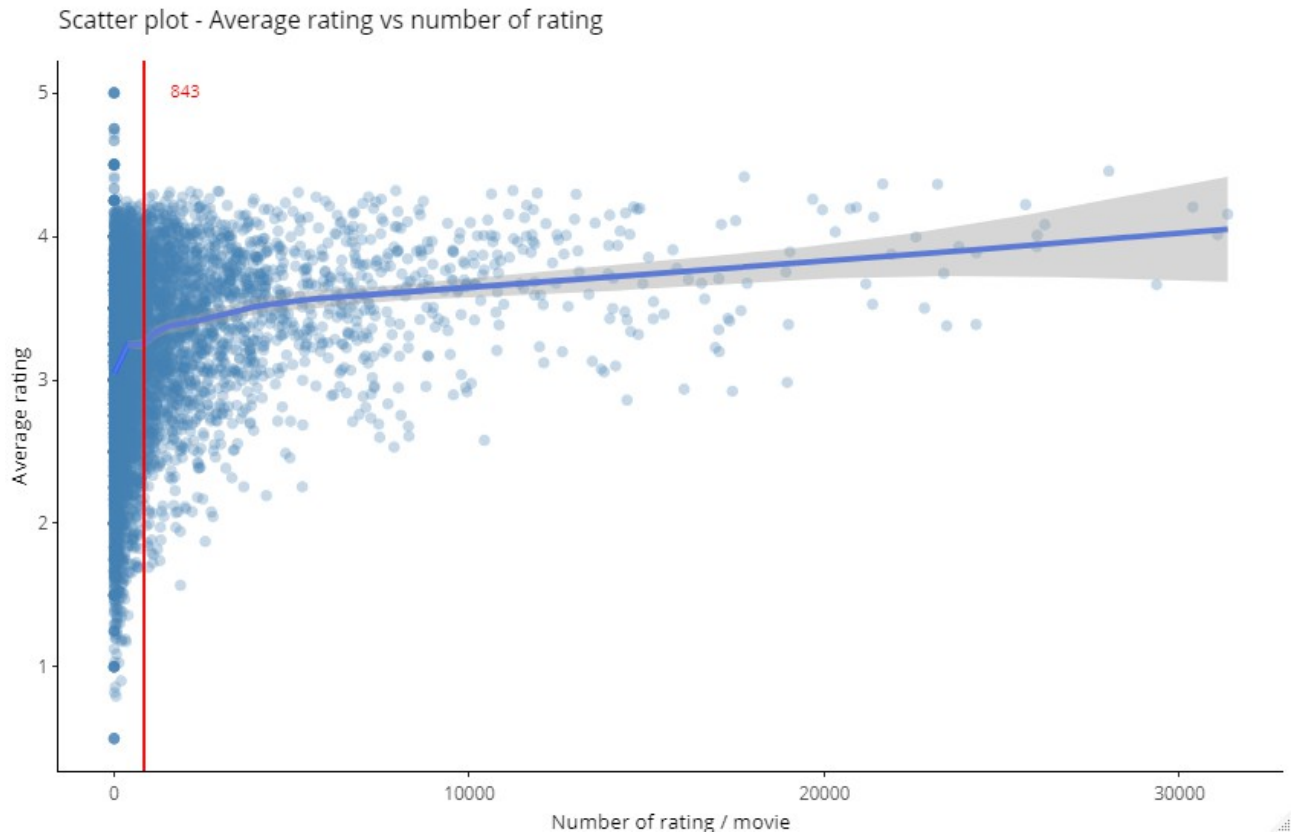
```
#create summary table grouping by movieId
movie_sum <- edx %>% group_by(movieId) %>% summarize(n_rating_of_movie = n(),  mu_movie = mean(rating), sd_movie = sd(rating)) #create figure of number of rating gg <- movie_sum %>% ggplot(aes(n_rating_of_movie)) + geom_density(fill = "gold1") + labs(title = "Density plot - number of rating", x = "number of rating",  y = "density", caption = "Figure 3 - The long tail number of rating") + geom_vline(aes(xintercept = mean(movie_sum$n_rating_of_movie)), color = "red") + annotate("text", x = 2000, y = 0.0022,  label = print(round(mean(movie_sum$n_rating_of_movie),0)),  color = "red", size = 3) + theme_classic() + theme(axis.title.x = element_text(size = 10), axis.title.y = element_text(size = 10), plot.title = element_text(size = 12), legend.position = "none")  ggplotly(gg)
```



We can see in figure 3 the number of rating per movie is vary from 1 to 31362, while in average a movie get only 843 ratings by user. There are 8553 movies, approximate 80 % number of movies, have number of rating less than 843 but only contribute 15 % number of rating. The remain 2124 movies, which number of rating is greater than 843, contribute 85 number of rating in whole data set. Prediction accuracy for movies with less number of rating is probably a challenge. The first ratings of those movies, probably come from its production, owners, respective people. In the edx set, there are 5% movies with less than 5 number of ratings, 9.9% movies with less than 10 number of ratings. This is not a high ratio.

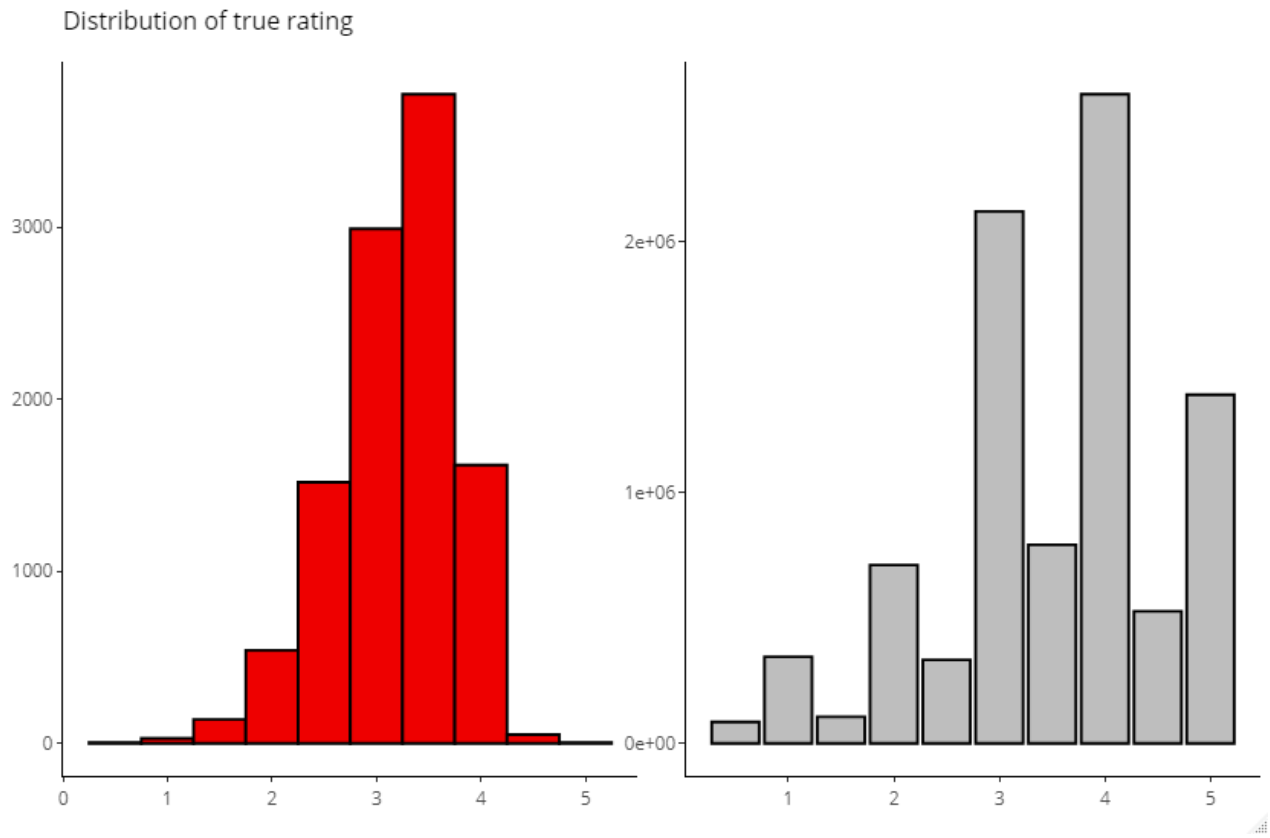
Another perspective from the figure below, this first one is rating of the movies with less number of rating are widely varied from 0.5 star to 5 stars, white rating of the movies with high number of rating are more focused. This is probably a challenge to predict the true rating for movie with very less number of rating. The second one is average rating is increased when the number of rating in increasing.

```
gg <- movie_sum %>%  ggplot(aes(n_rating_of_movie, mu_movie)) + geom_point(color = "steel blue", alpha = 0.3) + geom_smooth()+ geom_vline(aes(xintercept = mean(movie_sum$n_rating_of_movie)), color = "red")+ annotate("text", x = 2000, y = 5, label = print(round(mean(movie_sum$n_rating_of_movie),0)),  color = "red", size = 3) + theme_classic() + labs(title = "Scatter plot - Average rating vs number of rating", x = "Number of rating / movie", y = "Average rating", caption = "Figure 4") + theme(axis.title.x = element_text(size = 10), axis.title.y = element_text(size = 10), plot.title = element_text(size = 12))
ggplotly(gg)
```



The next figure is about the distribution of average rating by movieId.

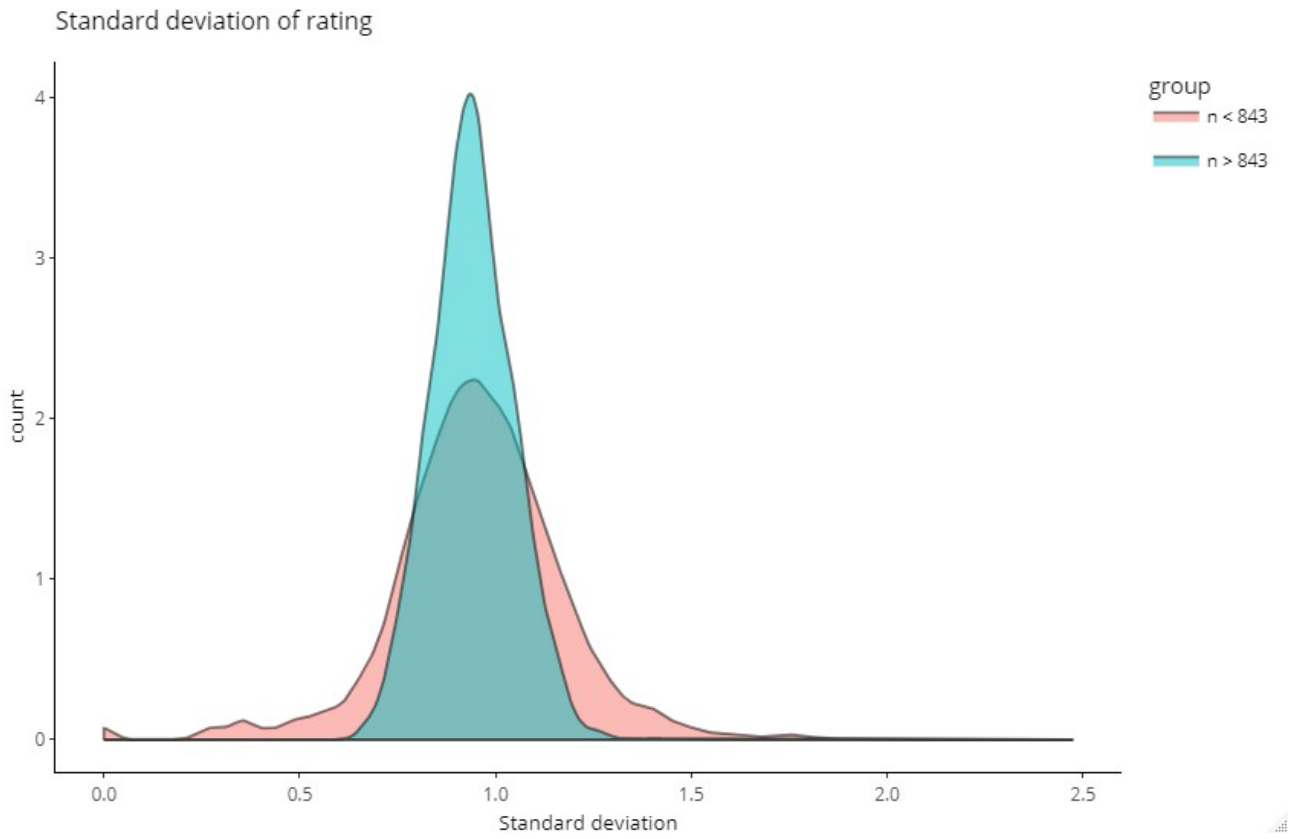
```
subplot(ggplotly(movie_sum %>% ggplot(aes(mu_movie)) + geom_histogram(fill = "red2", color = "black", binwidth = 0.5) + labs(title = "Distribution of movie's average rating", x = "rating",  y = "count", caption = "Figure 5") + theme_classic() + theme(axis.title.x = element_text(size = 10), axis.title.y = element_text(size = 10), plot.title = element_text(size = 12))), ggplotly(rating_sum %>% ggplot(aes(x = rating, y = count)) + geom_col(fill = "grey", color = "black") + labs(title = "Distribution of true rating", x = "rating", y = "count", caption = "Figure 6") + theme_classic() + theme(axis.title.x = element_text(size = 10), axis.title.y = element_text(size = 10), plot.title = element_text(size = 12))), nrows = 1)
```



We can see the number of true rating at 5 stars is extremely higher than the average rating at 5 stars in the left figure. The rating at 5 stars is approximate 15.45% of total rating, while only 0.12% movie have averating greater than 4.5 stars. Practically, some users are tend to rate at 5 stars if they are a big fan of a movie, or fan of an actor/actress in the movie, or because they always given 5 stars for their favorite genres, or just because of their habit (alway give 5 stars for any movie).

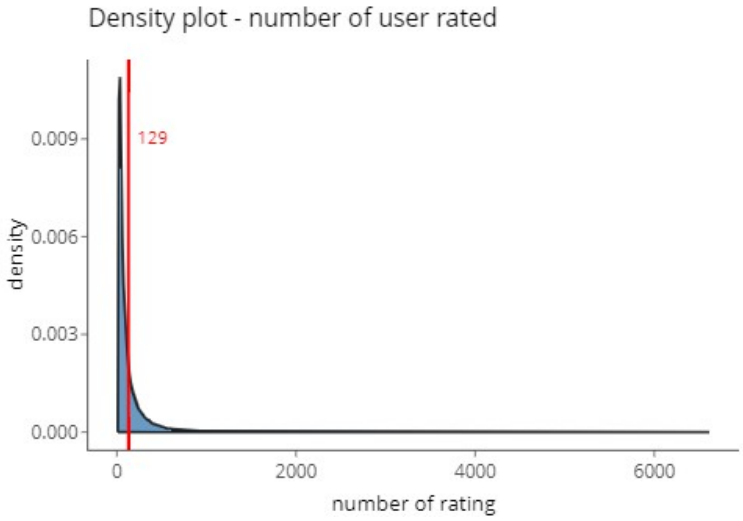
Quality of a movie, which is defined from user's perspective whom watched this movie, and represented by the rating given by those users. In the next figure, which is highlighted the density of the standard deviation of group movies with the number of rating less than vs greater than 843 (the average value number of rating per movie), is validated for the observation that the rating is less varied when the number of rating is increasing.

```
gg <- movie_sum %>% mutate(group = cut(n_rating_of_movie, breaks = c(-Inf, mean(n_rating_of_movie),Inf), labels = c("n < 843", "n > 843"))) %>% ggplot(aes(sd_movie, fill = group)) + geom_density(alpha = 0.5) + labs(title = "Standard deviation of rating", x = "Standard deviation", y = "count", caption = "Figure sd -  N < 843 number of rating less than average,  N > 843 number of rating greater than average") +  theme_classic() + theme(axis.title.x = element_text(size = 10), axis.title.y = element_text(size = 10), plot.title = element_text(size = 12))
```



Users

```
#create summary table grouping by userId
user_sum <- edx %>% group_by(userId) %>% summarize(n_user_rated = n(), mu_user = mean(rating),  sd_user = sd(rating))
#create figure of number of rating
gg <- user_sum %>% ggplot(aes(n_user_rated)) + geom_density(fill = "steel blue", alpha = 0.8) + labs(title = "Density plot - number of user rated", x = "number of rating", y = "density", caption = "Figure 8") + geom_vline(aes(xintercept = mean(user_sum$n_user_rated)), color = "red")+ annotate("text", x = 400, y = 0.009, label = print(round(mean(user_sum$n_user_rated),0)), color = "red", size = 3) + theme_classic() + theme(axis.title.x = element_text(size = 10), axis.title.y = element_text(size = 10), plot.title = element_text(size = 12), legend.position = "none")
ggplotly(gg)
```



A same density plot number of rating given by each users in figure 8, 30 % users have the number of rating given higher than the average of all users, contribute over 70 %number of rating. This mean that some users are very active than other users.

The top 10 users have highest number of rating as as below:

```
#top 10 users have highest number of rating
user_sum %>% arrange(desc(n_user_rated)) %>% head(10) %>% kable(caption = "Top 10 users with highest number of rating given", digits = 2) %>% kable_styling(font_size = 10, position = "center", latex_options = "HOLD_position")
```

Top 10 users with highest number of rating given

userId	n_user_rated	mu_user	sd_user
59269	6616	3.26	0.64
67385	6360	3.20	0.96
14463	4648	2.40	0.69
68259	4036	3.58	1.05
27468	4023	3.83	0.73
19635	3771	3.50	0.78
3817	3733	3.11	0.58
63134	3371	3.27	0.96
58357	3361	3.00	0.80
27584	3142	3.00	0.72

Compare to the top 10 users with lowest number of rating given.

```
#top 10 users have lowest number of rating
user_sum %>% arrange(n_user_rated) %>% head(10) %>% kable(caption = "Top 10 users with lowest number of rating given", digits = 2) %>% kable_styling(font_size = 10, position = "center", latex_options = "HOLD_position")
```

Top 10 users with lowest number of rating given

userId	n_user_rated	mu_user	sd_user
62516	10	2.25	1.14
22170	12	4.00	0.74
15719	13	3.77	1.24
50608	13	3.92	1.44
901	14	4.71	0.47
1833	14	3.00	1.24
2476	14	2.93	1.33
5214	14	1.79	1.22
9689	14	3.57	1.22
10364	14	4.32	1.27

In average, an user given 129 ratings, while the highest number of rating given by a user was 6616 and the lowest number of rating given by a user was 10. This is very unbalance.

Genres

Genre is the term for any category of literature or other forms of art or entertainment, e.g. music, whether written or spoken, audio or visual, based on some set of stylistic criteria. Genres are formed by conventions that change over time as new genres are invented and the use of old ones are discontinued. Often, works fit into multiple genres by way of borrowing and recombining these conventions. A movie could be classified to one or more genres, grouping by variable genres in the edx data set give us 797 levels of genres. If we separated it out, there are only 20 levels of genre.

```
#create a vector of genres
genres <- str_replace(edx$genres,"\\|.*","")
genres <- genres[!duplicated(genres)]
genres
```

```
##  [1] "Comedy"             "Action"             "Children"
##  [4] "Adventure"          "Animation"          "Drama"
##  [7] "Crime"              "Sci-Fi"             "Horror"
## [10] "Thriller"           "Film-Noir"          "Mystery"
## [13] "Western"            "Documentary"        "Romance"
## [16] "Fantasy"            "Musical"            "War"
## [19] "IMAX"               "(no genres listed)"
```

Also, we calculate the number of movies per each genre and average rating per each genres.

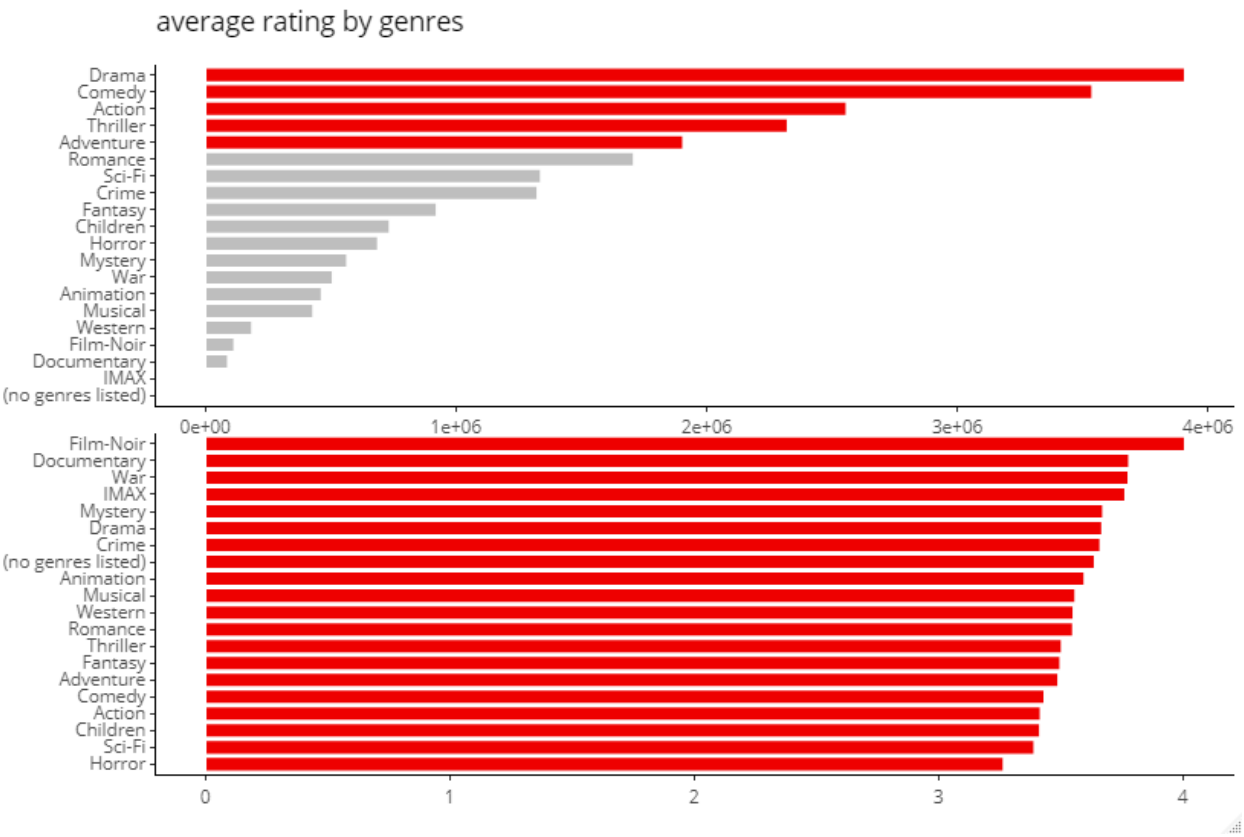
```
#calculate the number of movies per each genres
n_genres <- sapply(genres, function(ge){
index <- str_which(edx$genres, ge)
length(edx$rating[index])
})
#calculate the average rating of each genres
genres_rating <- sapply(genres, function(ge){
index <- str_which(edx$genres, ge)
mean(edx$rating[index], na.rm = T)
})

#create a summary data by genres
genres_sum <- data.frame(genres = genres, n_genres = n_genres, average_rating = genres_rating)

#print out the summary table by genres
genres_sum %>% arrange(desc(n_genres)) %>% head %>% kable(caption = "Summary table by genres", digits = 2) %>%
kable_styling(font_size = 10, position = "center", latex_options = "HOLD_position")
```

Summary table by genres

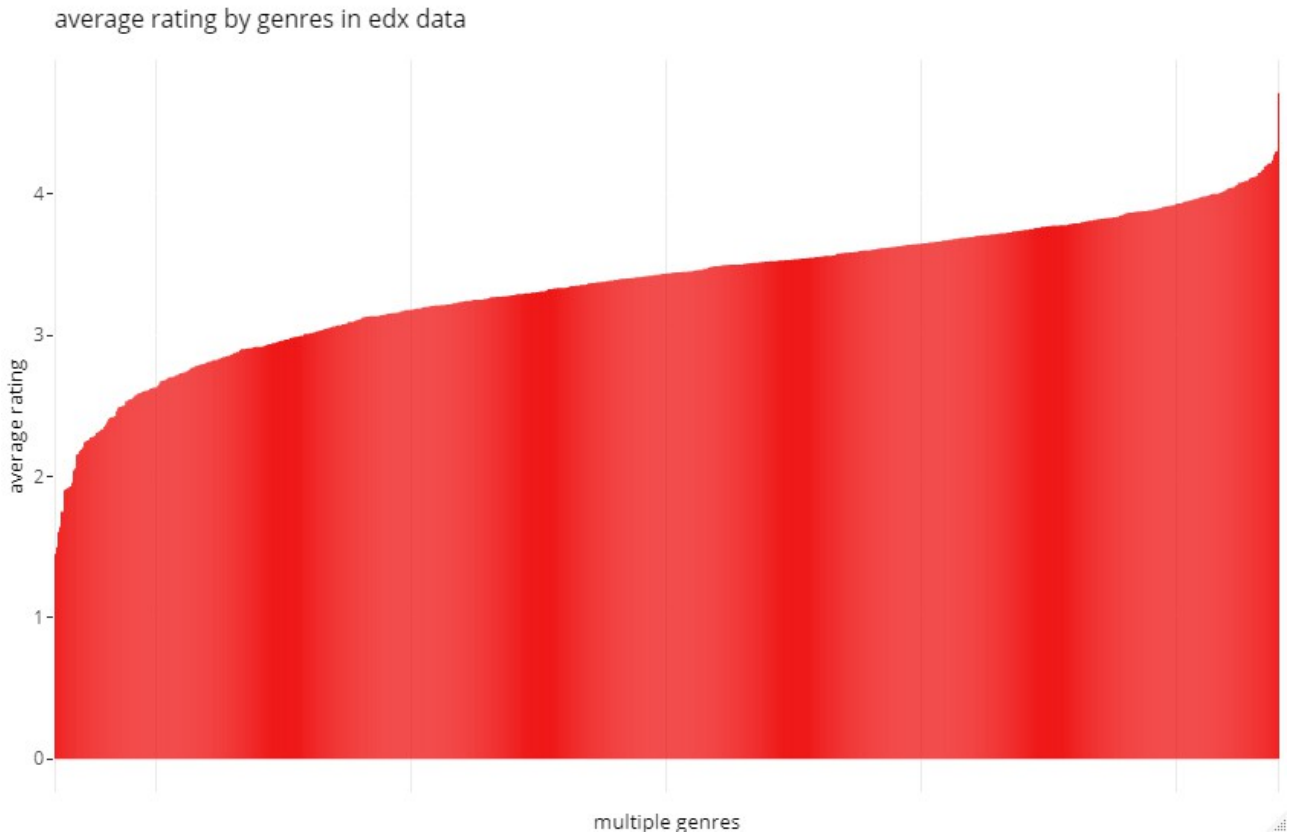
genres	n_genres	average_rating
Drama	3910127	3.67
Comedy	3540930	3.44
Action	2560545	3.42
Thriller	2325899	3.51
Adventure	1908892	3.49
Romance	1712100	3.55

Now we can see the average rating by genres.

```
subplot(
 #ranking genres by number of each appear in the edx data set
genres_sum %>% mutate(top5 = ifelse(genres %in% c("Comedy","Drama","Action","Thriller","Adventure"),
"top5","non")) %>% ggplot(aes(x = reorder(genres, n_genres), n_genres, fill = top5)) + geom_col(color = "white") +
theme_classic() + coord_flip() + labs(title = "number of movie by genres", y = "number of rating", x = "genres", caption
= "Figure 15") + scale_fill_manual(values = c("grey","red2")) + theme(legend.position = "none"),
# comparing average rating of each genres in edx data set
ggplot(genres_sum,  aes(x = reorder(genres, average_rating), average_rating)) + geom_col(fill = "red2", color =
"white") + theme_classic() + coord_flip() + labs(title = "average rating by genres", y = "average rating", x = "genres",
caption = "Figure 16"), nrows = 2)
```



Top 5 popular genres Drama, Comedy, Action, Thriller, Adventure appear 61 % in the edx data set. *Film-Noir* have highest average rating and *Horror* have lowest average rating compared to other movies. Because almost movies have more than 1 genre, we will analyze their rating change by original multiple genres instead of single genre. Similarly, average rating is varied across movie genres. Suppose this variation is due to user effect, we will select a group of users to validate this hypothesis. The users is sampling based on the number of rating they given, to choose group of average users, the number of rating given is approximate the average value 129 +/- 1. Similarly, the average rating is varied across genres, and we can see the variation is bigger, some genres are rated at 5 stars, and some genres are rated at 1 star.



Model analysis “RMSE”

The Central Limit Theorem (CLT) tells us that when the sample size is large enough, the probability distribution of the sum of the independent sample is approximately normal. Large is a relative term, however in many circumstances as few as sample size is more than 30 is enough to make the CLT useful. In some specific instances, as few as 10 is enough. By the law of the large number, we know that the standard error of the average becomes smaller and smaller as the sample size grows larger. When the sample size is very large, then the standard error is practically 0 and the average of the sample converges to the average of the population. Therefore when a movie have the number of rating increased, the overall rating of the movie would be closed to the true rating. We assume that the true quality of movieimoviei represented by the average rating value μiμi. We develope our model which the baseline is the average rating of each movie.
Now we calculated the RMSE, effected only by the average rating of each movie (y^i,jy^i,j = μiμi):

```
model_1_movie <- RMSE(edx$mu_movie, edx$rating)
rmse_results <- data_frame(method="Only baseline is movie average", RMSE = model_1_movie)
rmse_results %>% kable(caption = "RMSE by method", digits = 4) %>% kable_styling(font_size = 10, position =
"center", latex_options = "HOLD_position")
```

RMSE is 0.9423, still far compared to our target.

Now, we add the specific-effect by user bjbj to improve our model accuracy. The specific-user effect can be calculated as following:
The bjbj is calculated by following code:

```
b_j_sum <- edx %>% mutate(yhat = rating - mu_movie) %>% group_by(userId) %>% summarize(n_user_rated = n(),
b_j = mean(yhat)
```

Adding this specific-user effect to our predicting model, the predicted ratings will be calculated as following:

```
edx <- edx %>% eft_join(b_j_sum, by = "userId") %>% mutate(mu_movie_user = mu_movie + b_j)
```

The RMSE is as following:

```
model_2_movie_user <- RMSE(edx$mu_movie_user, edx$rating)
rmse_results <- bind_rows(rmse_results, data_frame(method="Add Specific-effect of user",  RMSE =
model_2_movie_user)) rmse_results %>% kable(caption = "RMSE by method", digits = 4) %>%  kable_styling(font_size
= 10, position = "center", latex_options = "HOLD_position")
```

RMSE is 0.8567, we have increased our accuracy, we can still improve.

In our dataset, time is recorded as the Unix timestamp, which is a way to track time as a running total of seconds. This count starts at the Unix Epoch on January 1st, 1970 at UTC. Therefore, the Unix timestamp is merely the number of seconds between a particular date and the Unix Epoch.
To make this variable be more friendly, we convert this to date-time format and assigned it to new variable rating year.

```
edx <- edx %>% mutate(rating_time = as.Date(as.POSIXct(timestamp, origin = "1970-01-01"))) %>% mutate(rating_year = year(rating_time))
edx <- edx %>% mutate(release_year = as.integer(substr(title, str_length(title) – 4, str_length(title) - 1)))
release_year_sum <- edx %>% group_by(release_year) %>% summarize(n = n(), average_rating = mean(rating))
```

From our analysis we know that the average rating was decreasing when the aging time increased. Therefore we can study the relationship between the residual εu,iεu,i and remain variables which we did not add to our above model. We can decompose the true rating value as following:

```
b_time_sum <- edx %>% mutate(error = rating - mu_movie_user) %>% group_by(aging_time) %>% summarize(b_time = mean(error))
```

The predicted rating is calculated as following:

```
#calculate predicted rating
edx <- edx %>% left_join(b_time_sum, by = "aging_time")
edx$b_time[is.na(edx$b_time)] <- 0
edx <- edx %>% mutate(mu_movie_user_time = mu_movie_user + b_time)
model_3_movie_user_time <- RMSE(edx$mu_movie_user_time, edx$rating)
rmse_results <- bind_rows(rmse_results, data_frame(method="Movie - User - Time Effect Model",  RMSE = model_3_movie_user_time))
rmse_results %>% kable(caption = "RMSE by method", digits = 4) %>% kable_styling(font_size = 10, position = "center", latex_options = "HOLD_position")
```

Now the RMSE is 0.8563, with a little bit improvement.

From previous analysis, we know a user given higher rating for some genres (suppose those are his/her favor) and lower rating for some genres (not his/her favor) compared to other genres. By the same approach, we add the genres-effect and the true rating can be decomposed to:

```
#calculate the genres effect bias
b_genres_sum <- edx %>% mutate(error = rating - mu_movie_user_time) %>% group_by(genres) %>% summarize(b_genres = mean(error))
edx <- edx %>% left_join(b_genres_sum, by = "genres")
edx <- edx %>% mutate(mu_movie_user_time_genres = mu_movie_user_time + b_genres)
model_4_movie_user_time_genres <- RMSE(edx$mu_movie_user_time_genres, edx$rating)
rmse_results <- bind_rows(rmse_results, data_frame(method="Movie - User - Time - Genres Effect Model",  RMSE = model_4_movie_user_time_genres)) data.frame(rmse_results) %>% kable(caption = "RMSE by different method", digits = 4) %>% kable_styling(font_size = 10, position = "center", latex_options = "HOLD_position")
```

Now, our model reduce the RMSE to 0.856.

Results

Our final predicting model used 4 predictors, the baseline as average rating of each movie, the specific effect by each user, the time effect and genres effect. The predicted rating calculation formular is as following:

With:

y^i,j,time,genresy^i,j,time,genres the predicted rating value.
Miμi the baseline as average rating of movie i.
bjbj the specific effect of user j.
Btimebtime the effect of aging time.
Bgenresbgenres the effect of genres.

```
#calculate the rating time
validation <- validation %>%  mutate(rating_time = as.Date(as.POSIXct(timestamp, origin = "1970-01-01"))) %>%
mutate(rating_year = year(rating_time))
#calculate the aging time
validation <- validation %>% left_join(movie_sum, by = "movieId")
validation <- validation %>% mutate(aging_time = round((timestamp - first_rating_time)/60/60/24/30,0))
validation <- validation %>% left_join(b_j_sum, by = "userId") %>% left_join(b_time_sum, by = "aging_time") %>%
left_join(b_genres_sum, by = "genres")
kable(data.frame(n_NA = colSums(is.na(validation[,14:16]))), caption = "NA value check in validation set", digits = 0)
%>%  kable_styling(font_size = 10, position = "center", latex_options = "HOLD_position")
```

We will replace the NA value in btimebtime by the average.

```
validation$b_time[is.na(validation$b_time)] <- mean(validation$b_time, na.rm = T)
validation <- validation %>% mutate(predicted_rating = mu_movie + b_j + b_time + b_genres)
```

The RMSE is as following:

```
RMSE(validation$rating, validation$predicted_rating)
```

The RMSE is 0.8645 less than our target 0.8650 as well as the the required RMSE to get the maximum point for the EDX Capstone Movielens projects (required RMSE ≤≤ 0.8649).

Final Validation

Recommender systems use historical data to make predictions. One of the most popular approaches in achieving this is collaborative filtering. It is based on historical behavior by its users. So far we have approached a dataset that features sparsity and biases with models that account for these effects with decent accuracy. To get better results we turn to a more advanced method called matrix factorization. Our user data is processed as a large and sparse matrix, then decomposed into two smaller dimensional matrices with latent features and less sparsity. To make the process more efficient the recosystem packgage will be used. We start by converting data into the recosystem format, find the best tuning parameters, train and finally test it.

```
library(recosystem)
set.seed(1, sample.kind="Rounding")
train_reco <- with(train_set, data_memory(user_index = userId, item_index = movieId, rating = rating))
test_reco <- with(test_set, data_memory(user_index = userId, item_index = movieId, rating = rating))
r <- Reco()
para_reco <- r$tune(train_reco, opts = list(dim = c(20, 30),  costp_l2 = c(0.01, 0.1), costq_l2 = c(0.01, 0.1), lrate =
c(0.01, 0.1),  nthread = 4, niter = 10))
r$train(train_reco, opts = c(para_reco$min, nthread = 4, niter = 30))
results_reco <- r$predict(test_reco, out_memory())
```

With the algorithm trained we now test it to see the resulting RMSE:

```
factorization_rmse <- RMSE(results_reco, test_set$rating)
results <- bind_rows(results, tibble(Method = "Model 5: Matrix factorization using ricosystem", RMSE =
factorization_rmse))
results %>% knitr::kable()
```

The final RMSE is 0.7843971.

Having found our model with the lowest RMSE using matrix factorization, the final step is to train it using the edx set and then test its accuracy on the validation set:

```
set.seed(1, sample.kind="Rounding")
edx_reco <- with(edx, data_memory(user_index = userId, item_index = movieId, rating = rating))
validation_reco <- with(validation, data_memory(user_index = userId, item_index = movieId, rating = rating))
r <- Reco()
para_reco <- r$tune(edx_reco, opts = list(dim = c(20, 30), costp_l2 = c(0.01, 0.1), costq_l2 = c(0.01, 0.1), lrate = c(0.01,
0.1), nthread = 4, niter = 10))
r$train(edx_reco, opts = c(para_reco$min, nthread = 4, niter = 30))
final_reco <- r$predict(validation_reco, out_memory())
final_rmse <- RMSE(final_reco, validation$rating)
results <- bind_rows(results, tibble(Method = "Final validation: Matrix factorization using recosystem", RMSE =
final_rmse)) results %>% knitr::kable()
```

Our final RMSE is 0.7811. Significantly below our target of 0.8649. We built and tested several models and achieved our best accuracy using matrix factorization which was simplified through the recosystem package.

Conclusions

In this report we described a way to build up the recommendation algorithm to predict movie ratings using Movielens data set step by step. Four predictors were used in our algorithm included: (i) the baseline as average rating of each movie, (ii) the specific-effect by user, (iii) the specific-effect by aging time, (iv) the specific-effect by genres. Two main predictors have highest impact to the results are the average rating of each movie and the specific-effect by user. The final RMSE from our algorithm is 0.7811. This result achieved our project goals and the initial criteria of the course HarvardX - PH125.9X Data Science: Capstone project - All learners (RMSE ≤≤0.7811).
Because our algorithm is based on the average rating of each movie, therefore if a movie have very less number of rating, this algorithm is limited to provide an accuracy results. A better result is received only when the number of rating of a movie and/or the number of rating given by a user is increased high enough.

However this is an opportunity to improve our algorithm in the future by including the number of rating of a movie and the number of rating given by a user to our algorithm. Furthermore, other machine learning techniques such as Regularization, Penalized Least Squares, Matrix Factorization could also improve the results further.

Another limitation during this project time is the hardware, especially RAM as the main constraint, and the speed of normal laptop.