MovieLens Project

Ng Da Xuan

5/31/2020

INTRODUCTION

The dataset used in this project came from the work of Harper & Konstan (2015). The dataset set is collected from a total of 71567 users, consists of 10000054 ratings, 95580 tags, and 10681 movies. The goal of the current project is to develop a model for the dataset, with prediction that results in low RMSE.

The key steps taken to reach the goal were: 1) dividing dataset into 'edx' dataset and 'validation' dataset, 2) exploring the 'edx' dataset and understanding the data structure, 3) identifying key variables, 4) sub-dividing 'edx' dataset into 'edx_train' set' and 'edx_test' set, 5) selecting the best linear combination model from the relevant variables, & finally 6) building a regularised linear model. The final regularised linear model is used to predict RMSE in the 'validation' set.

The following are the libraries used in exploring the data.

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

DATA PREPARATION

Step 1: Downloading data and dividing dataset into 2 ('edx' and 'validation')

- Download data

```
dl <- tempfile()
download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)
```

- Organised dataset into tidy format

```
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")
```

- Create 'validation' dataset and 'edx' dataset

```
## Validation set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

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

## Add rows removed from validation set back into edx set
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)
rm(dl, ratings, movies, test_index, temp, movielens, removed)
```

DATA DESCRIPTION

Step 2 & 3: Exploring 'edx' dataset and identifying key variables

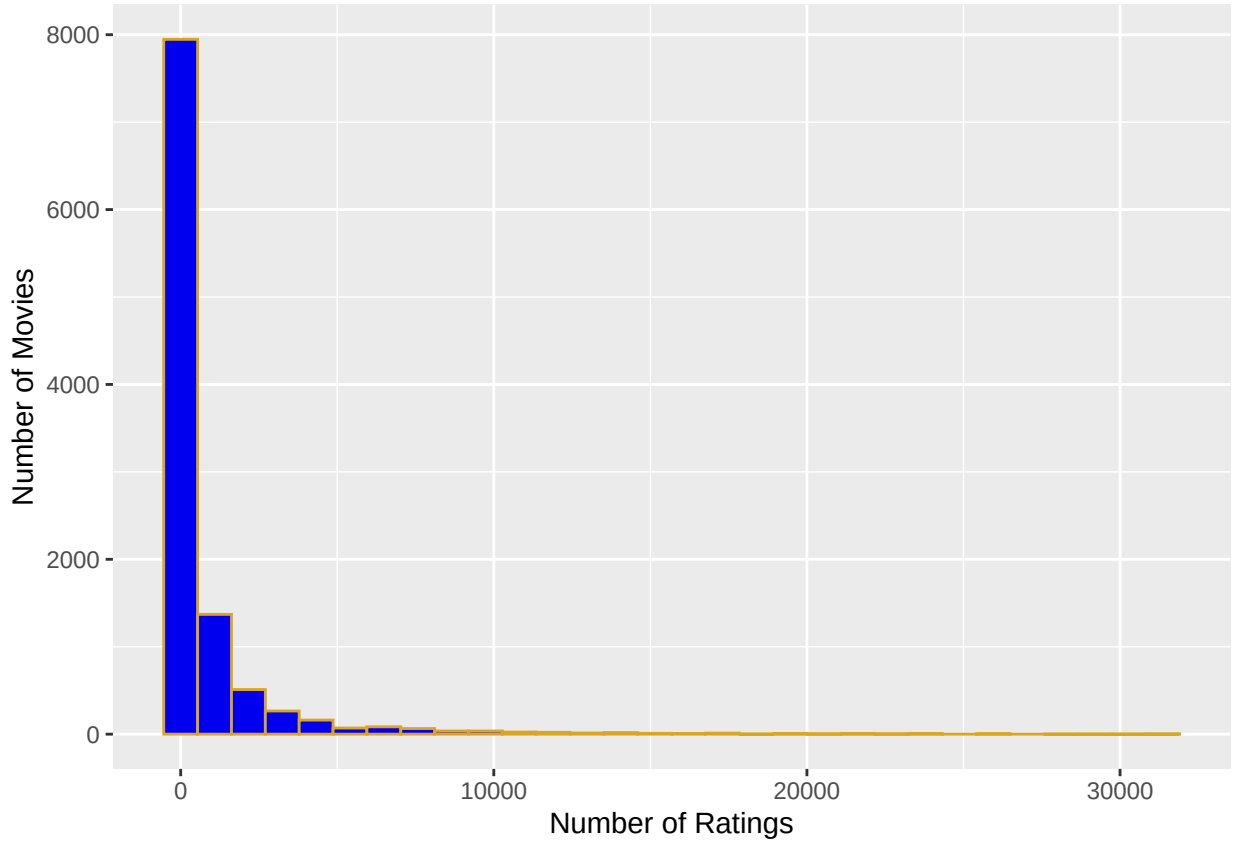
- Exploring the edx dataset

```
## Exploring the datatype and type of variables in the dataset.
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|A
```

```
## Number of unique movies and unique users in 'edx' dataset
edx%>%summarize(n_users = n_distinct(userId), n_movies = n_distinct(movieId))
```

```
##   n_users n_movies
## 1   69878    10677
```

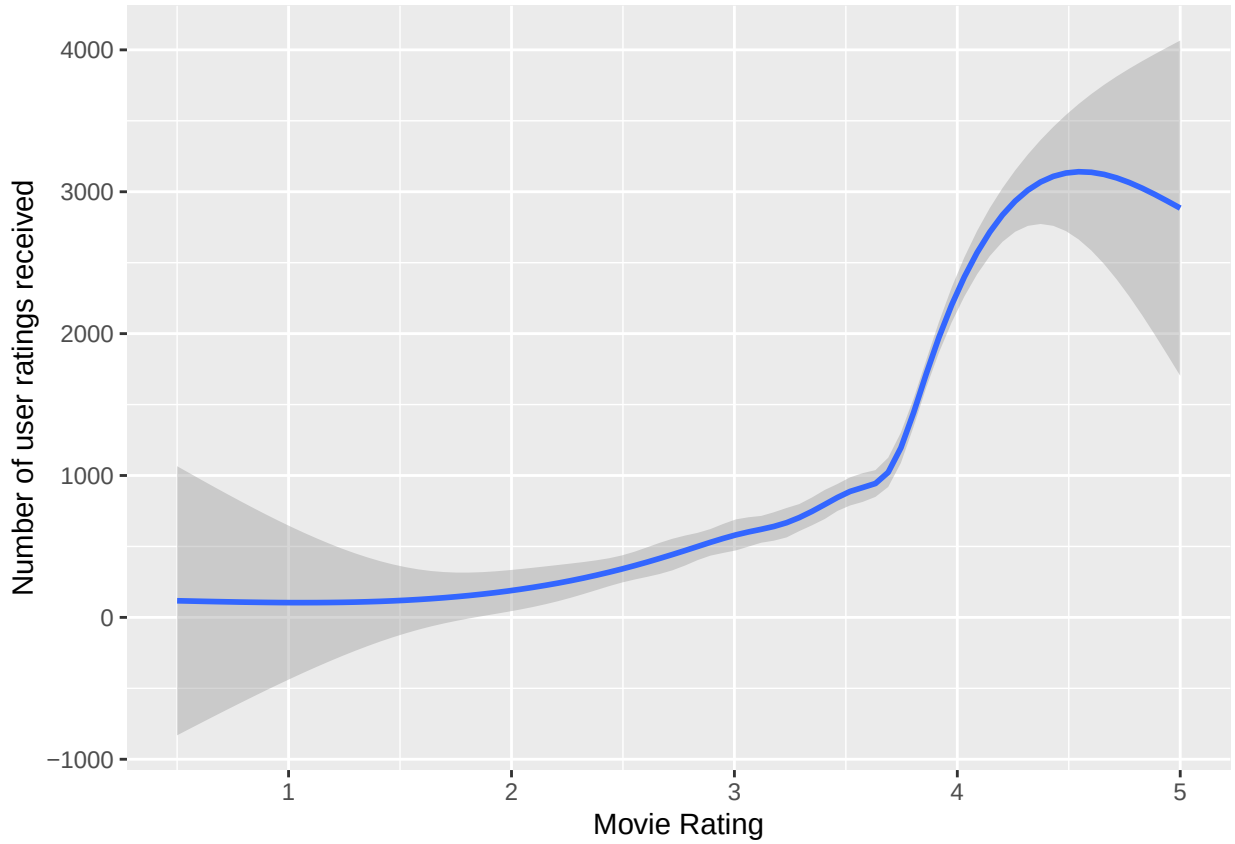


The above plot shows a positively skewed distribution, indicating that there are many movies with only few ratings.

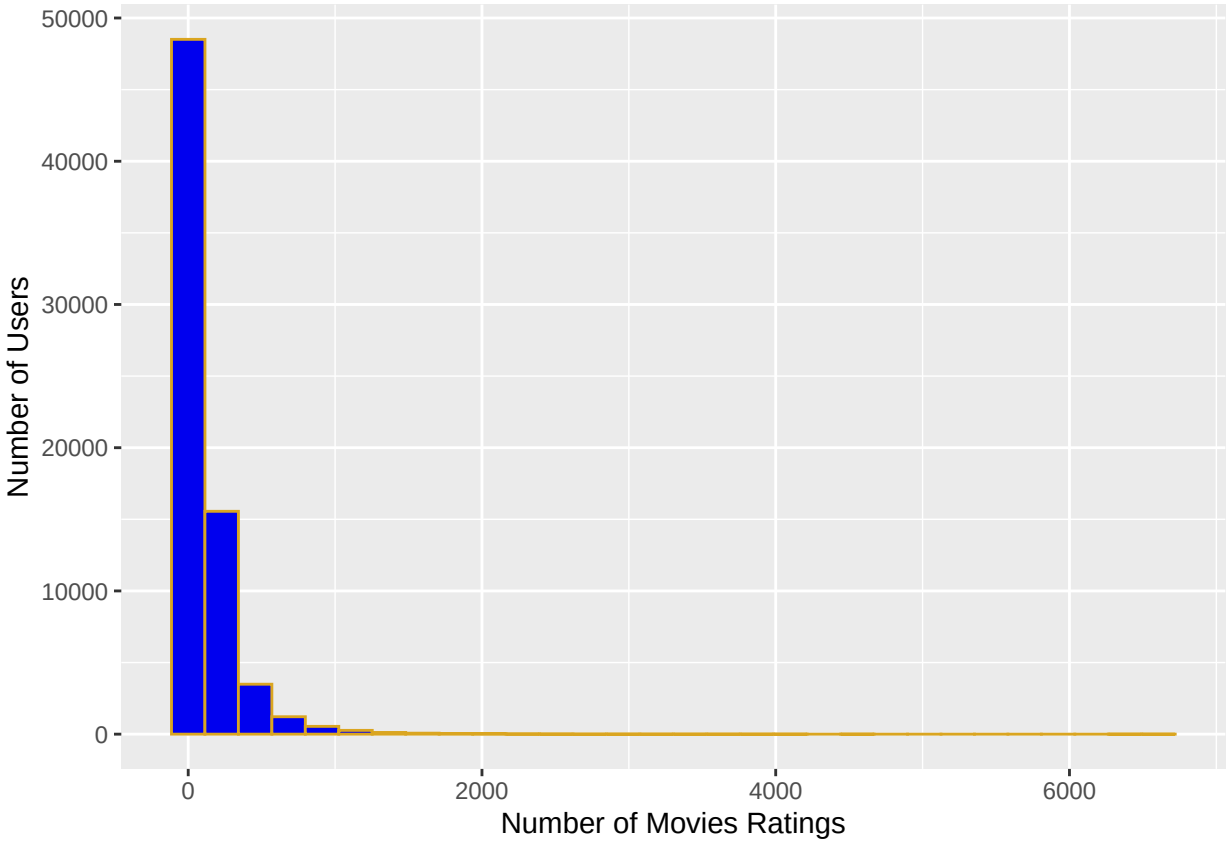
```
### Movie with high number of movie ratings appears to have higher movie rating
edx %>%
  group_by(movieId, title) %>%
  summarize(count = n(), rating = mean(rating)) %>%
  arrange(desc(count))%>%
  top_n(10, count)

### Movie with low number of movie ratings appears to have lower rating
edx %>% group_by(movieId, title) %>%
  summarize(count = n(), rating = mean(rating)) %>%
  arrange(count)%>%
  top_n(10, count)
```

```
## # A tibble: 10,677 x 4
## # Groups:   movieId [10,677]
##    movieId title                                                count rating
##      <dbl> <chr>                                                <int>  <dbl>
##  1     296 Pulp Fiction (1994)                                  31362   4.15
##  2     356 Forrest Gump (1994)                                  31079   4.01
##  3     593 Silence of the Lambs, The (1991)                     30382   4.20
##  4     480 Jurassic Park (1993)                                 29360   3.66
##  5     318 Shawshank Redemption, The (1994)                     28015   4.46
##  6     110 Braveheart (1995)                                    26212   4.08
```

```
##  7     457 Fugitive, The (1993)                                      25998   4.01
##  8     589 Terminator 2: Judgment Day (1991)                         25984   3.93
##  9     260 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (~ 25672   4.22
## 10     150 Apollo 13 (1995)                                          24284   3.89
## # ... with 10,667 more rows
## # A tibble: 10,677 x 4
## # Groups:   movieId [10,677]
##    movieId title                                                 count rating
##      <dbl> <chr>                                                 <int>  <dbl>
##  1    3191 Quarry, The (1998)                                        1    3.5
##  2    3226 Hellhounds on My Trail (1999)                             1    5
##  3    3234 Train Ride to Hollywood (1978)                            1    3
##  4    3356 Condo Painting (2000)                                     1    3
##  5    3383 Big Fella (1937)                                          1    3
##  6    3561 Stacy's Knights (1982)                                    1    1
##  7    3583 Black Tights (1-2-3-4 ou Les Collants noirs) (1960)       1    3
##  8    4071 Dog Run (1996)                                            1    1
##  9    4075 Monkey's Tale, A (Les ChÃ¢teau des singes) (1999)         1    1
## 10    4820 Won't Anybody Listen? (2000)                              1    2
## # ... with 10,667 more rows
```



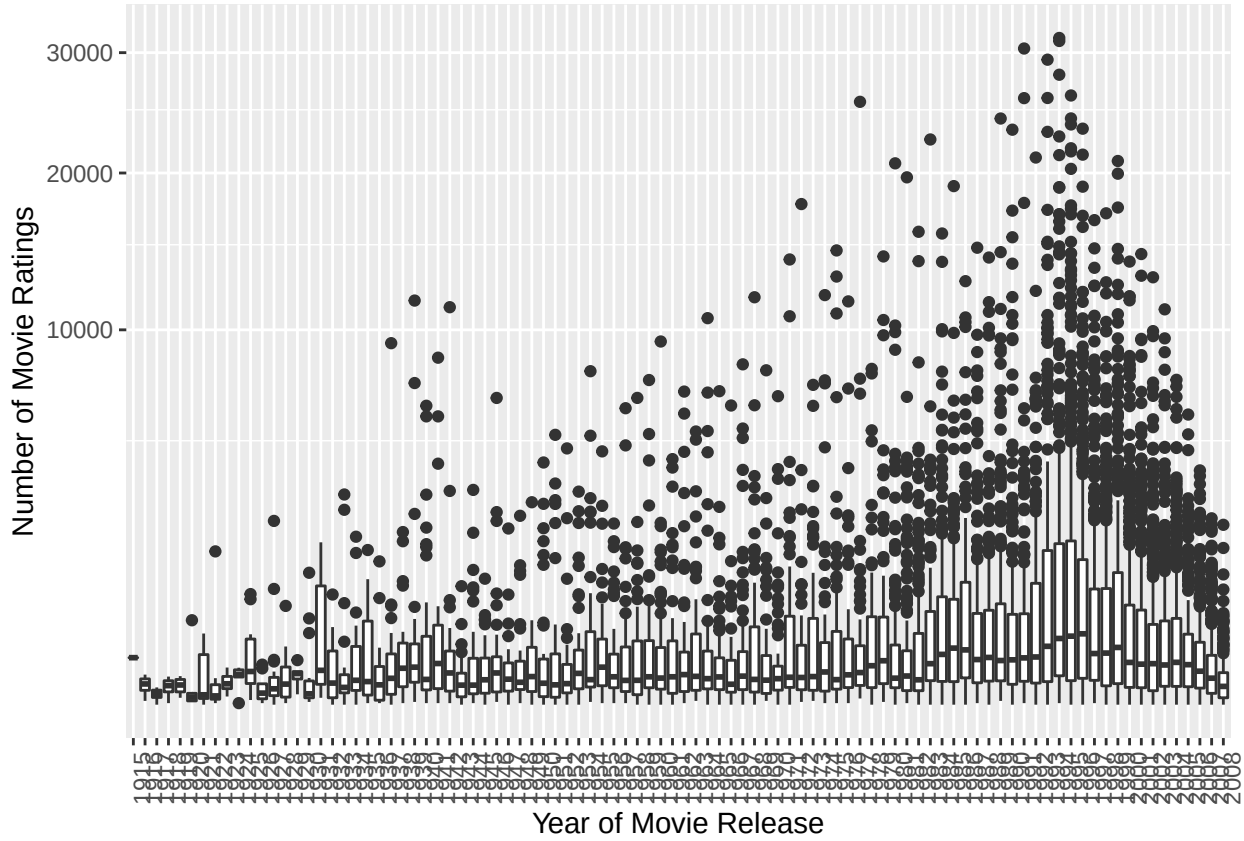
The above smooth curve shows that movie with better rating are likely to receive more reviews, and movies with poor rating are likely to receive few reviews. This shows a potential movie bias on rating.



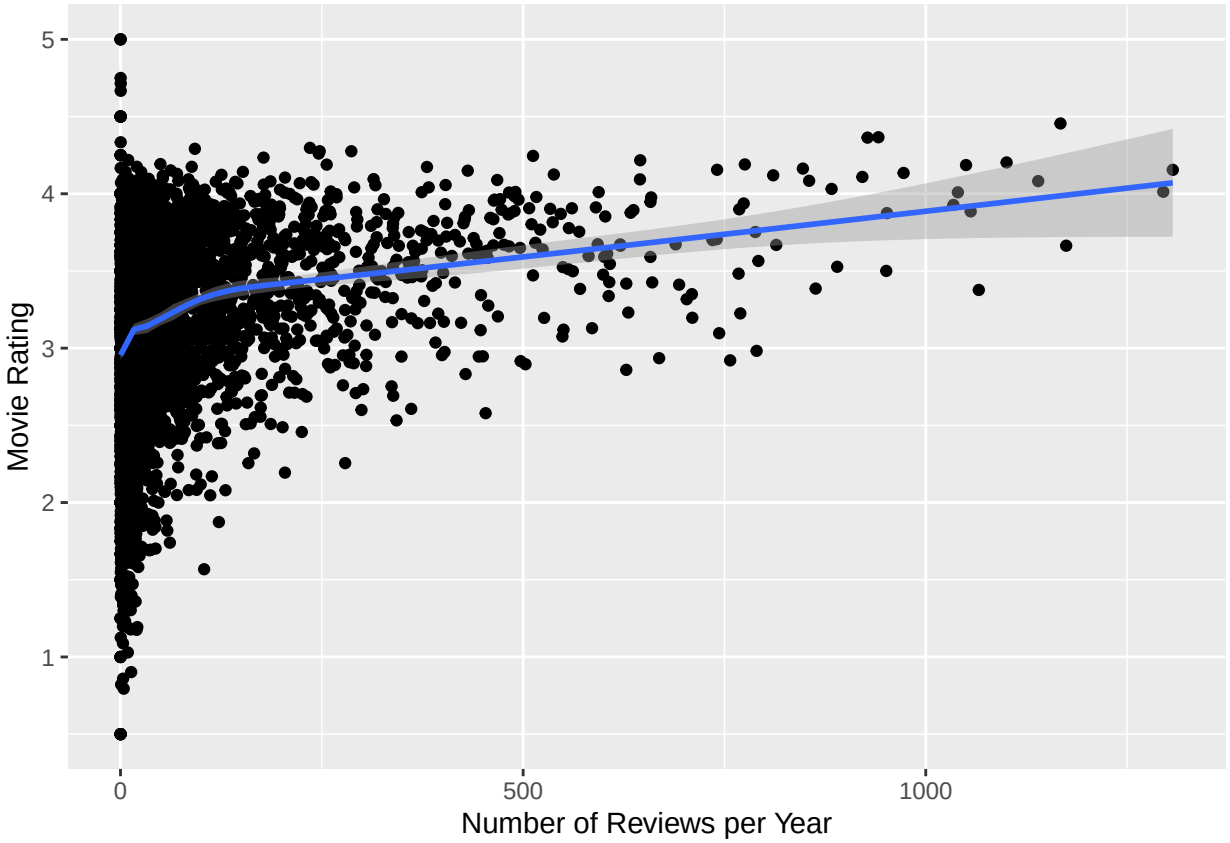
The above plot shows that most users give only few movie ratings.

To examine whether year of movie release has an impact on movie rating, movie title is parsed from the movie title, as shown below.

```
pattern<-"[(]+(\\d{4})[)]+$"
edx_with_year<- edx %>%
  mutate(release_year=parse_number(str_extract(title, pattern)))
```



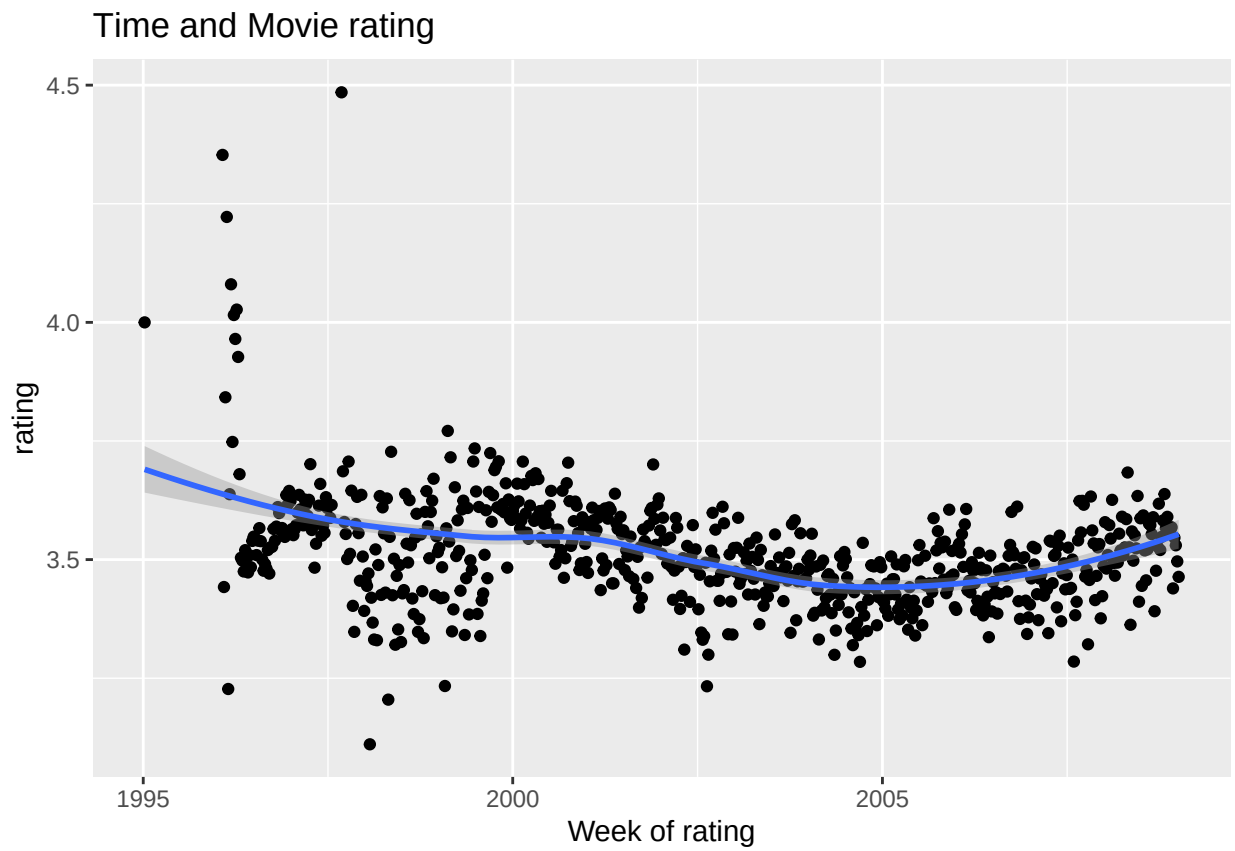
This graph shows that movies released in the 1990s received the most number of ratings.



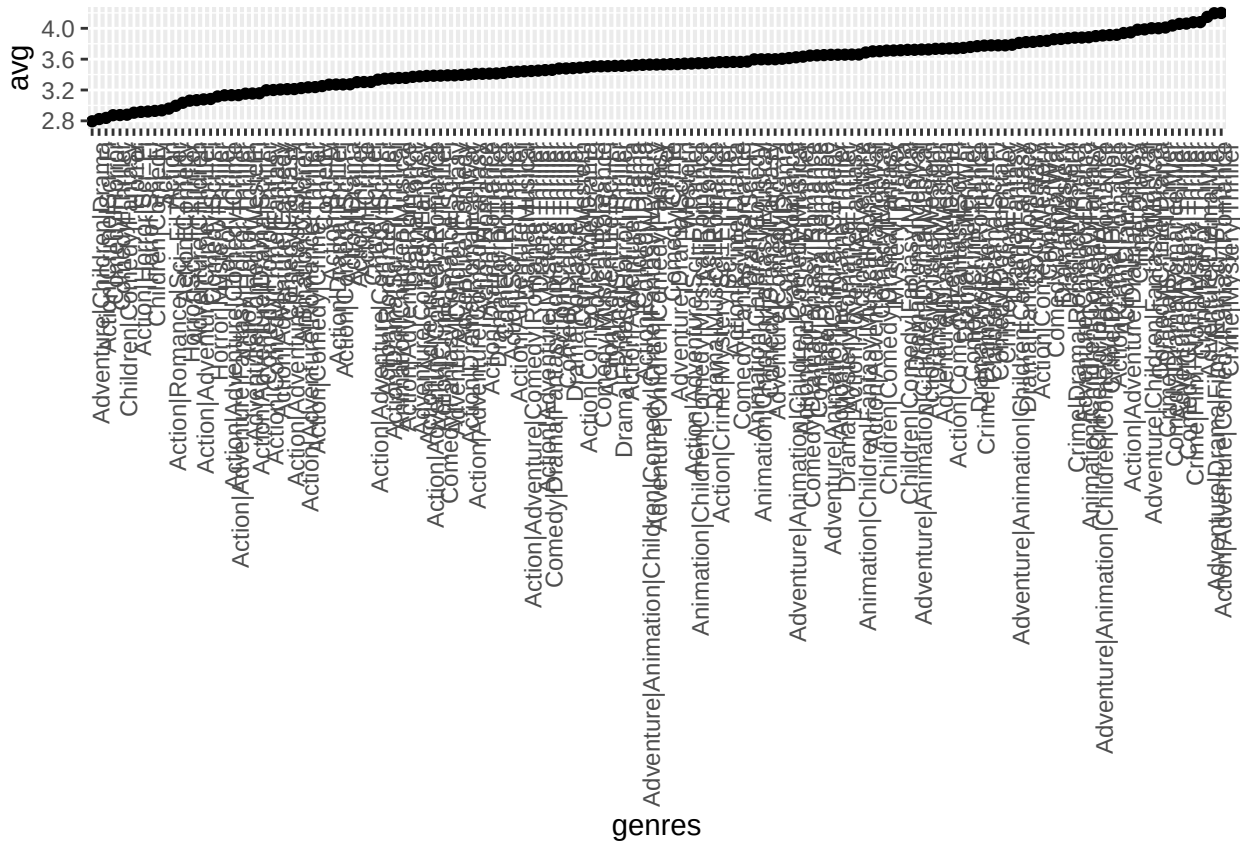
The above plot shows that movies with more review/year are likely to have higher movie ratings.

Others variables of interest are the timestamp of reviews (ie., rate_date), the number of years since the release of movie (ie., time_btw), the number of ratings received for each movie per year (ie., rate), and the total number of ratings received for each movie (ie., total_r). This variables are mutated from the 'edx' data, as shown below.

```
edx_with_year_date <- edx_with_year%>%
  mutate(date = as_datetime(timestamp))%>%
  mutate(rate_date = round_date(date, unit = "week"),
         time_btw = 2018 - release_year)

edx_with_year_week <- edx_with_year_date%>%
  group_by(movieId)%>%
  summarize(rate = n()/mean(time_btw),
            total_r = n())

edx_with_year_week_2 <- edx_with_year_date %>%
  left_join(edx_with_year_week, by='movieId')
```



This plot shows some change in movie rating as a result of time.



This plot shows that genres has an impact on movie rating. The exploration of data reveals 6 variables of interest: 1) movieId, 2) userId, 3) rate_date, 4) time_btw, 5) release_year, 6) number of ratings received, & 7) genres.

METHOD AND ANALYSIS

Step 4: Partitioning 'edx' dataset into 2 ('edx_train' set and 'edx_test') (to prevent overfitting)

```
test_index <- createDataPartition(y = edx_with_year_week_2$rating, times = 1, p = 0.5, list = FALSE)
edx_train <- edx_with_year_week_2[-test_index,]
temp <- edx_with_year_week_2[test_index,]

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

# Add rows removed from validation set back into edx set
removed <- anti_join(temp, edx_test)
edx_train <- rbind(edx_train, removed)
rm(test_index, temp,removed)
```

Step 5: Selecting the best linear combination model from the variables

First, create a RMSE function.

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

To examine the variable importance, each variable is examined individually. The first linear model is created with the mean of rating.

```
mu <- mean(edx_with_year_date$rating)
mu
```

```
## [1] 3.512465
```

```
just_mean_effect<-RMSE(mu, edx_test$rating)
just_mean_effect
```

```
## [1] 1.060407
```

The second model is created with just movie effect.

```
movie_ind_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))

predicted_ratings <- mu + edx_test %>%
  left_join(movie_ind_avgs, by='movieId') %>%
  pull(b_i)

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

```
## [1] 0.9441335
```

The third model is created with just user effect.

```
user_ind_avgs <- edx_train %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu))

predicted_ratings <- mu + edx_test %>%
  left_join(user_ind_avgs, by='userId') %>%
  pull(b_u)

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

```
## [1] 0.9820931
```

The fourth model is created with just year of release effect.

```
release_yr_ind_avgs <- edx_train %>%
  group_by(release_year) %>%
  summarize(b_rel_yr = mean(rating - mu))

predicted_ratings <- mu + edx_test %>%
  left_join(release_yr_ind_avgs, by='release_year') %>%
  pull(b_rel_yr)

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

```
## [1] 1.049341
```

The fifth model is created with just rate_date effect.

```
date_ind_avgs <- edx_train %>%
  group_by(rate_date) %>%
  summarize(b_dt = mean(rating - mu))


predicted_ratings <- mu + edx_test %>%
  left_join(date_ind_avgs, by='rate_date') %>%
  pull(b_dt)

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

```
## [1] 1.056729
```

The sixth model is created with just number of ratings received per year.

```
rate_ind_avgs <- edx_train %>%
  group_by(rate) %>%
  summarize(b_r = mean(rating - mu))

predicted_ratings <- mu + edx_test %>%
  left_join(rate_ind_avgs, by='rate') %>%
  pull(b_r)

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

```
## [1] 0.95016
```

The seventh model is created with just the genre effect.

```
genre_ind_avgs <- edx_train %>%
  group_by(genres) %>%
  summarize(b_g = mean(rating - mu))

predicted_ratings <- mu + edx_test %>%
```

```
  left_join(genre_ind_avgs, by='genres') %>%
  pull(b_g)

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

```
## [1] 1.018155
```

The eighth model is created with just the time between effect.

```
time_ind_avgs <- edx_train %>%
  group_by(time_btw) %>%
  summarize(b_time = mean(rating - mu))

predicted_ratings <- mu + edx_test %>%
  left_join(time_ind_avgs, by='time_btw') %>%
  pull(b_time)

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

```
## [1] 1.049341
```

The result of the individual model is summarised in the table below.

```
##                       method      RMSE
## 1          Just Movie Model 0.9441335
## 2 Just Rate per Year Model 0.9501600
## 3           Just User Model 0.9820931
## 4          Just Genre Model 1.0181551
## 5  Just Release Year Model 1.0493412
## 6  Just Between Time Model 1.0493412
## 7     Just Timestamp Model 1.0567292
## 8          Just the average 1.0604071
```

To build a linearised model, each variables are included into the model in order of importance.

We first examine movie + rate model.

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

rate_avgs <- edx_train %>%
  left_join(movie_avgs, by='movieId')%>%
  group_by(rate)%>%
  summarize(b_r = mean(rating - mu - b_i))

predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by = 'movieId')%>%
  left_join(rate_avgs, by = 'rate')%>%
  mutate(pred = mu + b_i + b_r)%>%
```

```
  pull(pred)

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

```
## [1] 0.9441335
```

There is no significant improvement in the "Movie + Rate' model when compared to the 'Movie only' Model.

Second, let's examine 'Movie + User' model.

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

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

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

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

```
## [1] 0.8696666
```

There is an visible improvement in 'Movie + User' model.

Next, we examined the model 'Movie + User + genre'

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

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

genres_avgs<-edx_train%>%
  left_join(movie_avgs,by='movieId')%>%
  left_join(user_avgs,by='userId')%>%
  group_by(genres)%>%
  summarize(b_g=mean(rating - mu - b_i - b_u))

predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by = 'movieId')%>%
  left_join(user_avgs, by = 'userId')%>%
```

```
  left_join(genres_avgs, by='genres')%>%
  mutate(pred = mu + b_i + b_u + b_g)%>%
  pull(pred)

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

```
## [1] 0.8693189
```

We add a fourth model ('Movie + User + Genre + Rate_date') to see if it helps improve the model.

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

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

genres_avgs<-edx_train%>%
  left_join(movie_avgs,by='movieId')%>%
  left_join(user_avgs,by='userId')%>%
  group_by(genres)%>%
  summarize(b_g=mean(rating - mu - b_i - b_u))

rate_date_avgs<-edx_train%>%
  left_join(movie_avgs,by='movieId')%>%
  left_join(user_avgs,by='userId')%>%
  left_join(genres_avgs, by='genres')%>%
  group_by(rate_date)%>%
  summarize(b_t=mean(rating - mu - b_i - b_u - b_g))

predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by = 'movieId')%>%
  left_join(user_avgs, by = 'userId')%>%
  left_join(genres_avgs, by='genres')%>%
  left_join(rate_date_avgs,by='rate_date')%>%
  mutate(pred = mu + b_i + b_u + b_g + b_t)%>%
  pull(pred)

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

```
## [1] 0.8692418
```

The results are summarised in the table below.

```
##                                    method      RMSE
## 1 Movie + User + Genres + Rate_date Model 0.8692418
## 2             Movie + User + Genres Model 0.8693189
## 3                      Movie + User Model 0.8696666
## 4                      Movie + Rate Model 0.9441335
```

Step 6: Building a regularised linear model

Due to limited computing resources (I am using my old computer with no access to good computers due to Covid-19 quarantine measures) and to prevent R from crashing, only the code for generating lambda is provided here. For more details, please refer to R script provided in Github.

To penalise for small sample size in both movieId and userId, lambda calculation is done for movieId and userId separately.

```
## 1) Finding movie lambda value (ie., the value is 4)
lambdas <- seq(3.5, 5, 0.25) # Multiple small brackets were used to isolate the lambda
#in order to save processing time. NOTE. Only code is provided here.

rmses <- sapply(lambdas, function(l){

  movie_avgs <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

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

  genres_avgs<-edx_train%>%
    left_join(movie_avgs,by='movieId')%>%
    left_join(user_avgs,by='userId')%>%
    group_by(genres)%>%
    summarize(b_g=mean(rating - mu - b_i - b_u))

  rate_date_avgs<-edx_train%>%
    left_join(movie_avgs,by='movieId')%>%
    left_join(user_avgs,by='userId')%>%
    left_join(genres_avgs, by='genres')%>%
    group_by(rate_date)%>%
    summarize(b_t=mean(rating - mu - b_i - b_u - b_g))

  predicted_ratings <- edx_test %>%
    left_join(movie_avgs, by = 'movieId')%>%
    left_join(user_avgs, by = 'userId')%>%
    left_join(genres_avgs, by='genres')%>%
    left_join(rate_date_avgs,by='rate_date')%>%
    mutate(pred = mu + b_i + b_u + b_g + b_t)%>%
    pull(pred)
  return(RMSE(predicted_ratings, edx_test$rating))
})

qplot(lambdas, rmses)
lambda_movie<-lambdas[which.min(rmses)]
```

```
## 2) Finding user lambda value (ie., the value is 5.25)

lambdas <- seq(4.75, 5.5, 0.25) # Multiple small brackets were used to isolate the lambda
# in order to save processing time. NOTE. Only code is provided here.
```

```
rmses <- sapply(lambdas, function(l){
  movie_avgs <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+lambda_movie))

  user_avgs <- edx_train %>%
    left_join(movie_avgs, by ='movieId')%>%
    group_by(userId)%>%
    summarize(b_u = sum(rating - mu - b_i)/(n()+l))

  genres_avgs<-edx_train%>%
    left_join(movie_avgs,by='movieId')%>%
    left_join(user_avgs,by='userId')%>%
    group_by(genres)%>%
    summarize(b_g=mean(rating - mu - b_i - b_u))

  rate_date_avgs<-edx_train%>%
    left_join(movie_avgs,by='movieId')%>%
    left_join(user_avgs,by='userId')%>%
    left_join(genres_avgs, by='genres')%>%
    group_by(rate_date)%>%
    summarize(b_t=mean(rating - mu - b_i - b_u - b_g))

  predicted_ratings <- edx_test %>%
    left_join(movie_avgs, by = 'movieId')%>%
    left_join(user_avgs, by = 'userId')%>%
    left_join(genres_avgs, by='genres')%>%
    left_join(rate_date_avgs,by='rate_date')%>%
    mutate(pred = mu + b_i + b_u + b_g + b_t)%>%
    pull(pred)

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

qplot(lambdas, rmses)
lambda_user <- lambdas[which.min(rmses)]
```

The lambda value for movie is 4, and the lambda value for userId is 5.25.

Let's calculate RMSE value for the regularised model using the obtained lambda values.

```
movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+4))

user_avgs <- edx_train %>%
  left_join(movie_avgs, by ='movieId')%>%
  group_by(userId)%>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+5.25))

genres_avgs<-edx_train%>%
  left_join(movie_avgs,by='movieId')%>%
  left_join(user_avgs,by='userId')%>%
```

```
  group_by(genres)%>%
  summarize(b_g=mean(rating - mu - b_i - b_u))

rate_date_avgs<-edx_train%>%
  left_join(movie_avgs,by='movieId')%>%
  left_join(user_avgs,by='userId')%>%
  left_join(genres_avgs, by='genres')%>%
  group_by(rate_date)%>%
  summarize(b_t=mean(rating - mu - b_i - b_u - b_g))

predicted_ratings <- edx_test %>%
  left_join(movie_avgs, by = 'movieId')%>%
  left_join(user_avgs, by = 'userId')%>%
  left_join(genres_avgs, by='genres')%>%
  left_join(rate_date_avgs,by='rate_date')%>%
  mutate(pred = mu + b_i + b_u + b_g + b_t)%>%
  pull(pred)

RMSE(predicted_ratings, edx_test$rating)
```

```
## [1] 0.8676022
```

FINAL RESULTS

First, we will mutate the validation dataset to obtain the "rate_date" variable.

```
validation_mutate <- validation %>%
  mutate(date = as_datetime(timestamp))%>%
  mutate(rate_date = round_date(date, unit = "week"))
```

Now, we will use the regularised model on the 'validation' dataset.

Results

```
mu <- mean(edx_train$rating)

movie_avgs <- edx_train %>%
  group_by(movieId) %>%
  summarize(b_i = sum(rating - mu)/(n()+4))

user_avgs <- edx_train %>%
  left_join(movie_avgs, by ='movieId')%>%
  group_by(userId)%>%
  summarize(b_u = sum(rating - mu - b_i)/(n()+5.25))

genres_avgs<-edx_train%>%
  left_join(movie_avgs,by='movieId')%>%
  left_join(user_avgs,by='userId')%>%
  group_by(genres)%>%
  summarize(b_g=mean(rating - mu - b_i - b_u))
```

```
rate_date_avgs<-edx_train%>%
  left_join(movie_avgs,by='movieId')%>%
  left_join(user_avgs,by='userId')%>%
  left_join(genres_avgs, by='genres')%>%
  group_by(rate_date)%>%
  summarize(b_t=mean(rating - mu - b_i - b_u - b_g))

predicted_ratings <- validation_mutate %>%
  left_join(movie_avgs, by = 'movieId')%>%
  left_join(user_avgs, by = 'userId')%>%
  left_join(genres_avgs, by='genres')%>%
  left_join(rate_date_avgs,by='rate_date')%>%
  mutate(pred = mu + b_i + b_u + b_g + b_t)%>%
  pull(pred)
```

```
RMSE(predicted_ratings, validation_mutate$rating)
```

```
## [1] 0.867512
```

Conclusion

The model adequately predicted the validation dataset. That is, the model predict more than 'Just Average' model.

The RMSE obtained from this model can be improved much further if the train function of caret package can be utilised. For example, a function distribution can be created for time effect and genre effect. Unfortunately, due to the author's limited computing resources, knn and loess model is not being examined.

We recommend for further work by using knn or loess model on the time and genre distribution to predict movie rating.

References

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872