MovieLens Project

Masahiro TAKAOKA

2020/1/7

Overview

This project is part of HarvardX: PH125.9x Data Science: Capstone course and the purpose of this project is to predict user reviews for movies. This report includes not only prediction but also exploratory data analysis, understanding the uniqueness of the data and searching for a machine learning model suitable for the task.

Note: If the result is not returned due to heavy processing, please run again or change the equipment.

Evaluation

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

The closer the RMSE is to 0, the smaller the estimated prediction error, that is, the higher the prediction accuracy.

Dataset

The URL of the data set to be used is as follows. https://grouplens.org/datasets/movielens/10m/
http://files.grouplens.org/datasets/movielens/ml-10m.zip

Data Loading and Create Train and Validation Sets

There is an instruction from edx in advance about loading and splitting the dataset, and the following code is also provided.

```
################################
# Create edx set, validation set
################################

# Note: this process could take a couple of minutes

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")
if(!require(lubridate)) install.packages("lubridate", 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"))
```

```r
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")

# Decomposes timestamp stored in UNIX time into year, month, and day units
movielens <- movielens %>%
  mutate(timestamp_year = format(as.POSIXct(timestamp, origin="1970-1-1"), format="%Y")) %>%
  mutate(timestamp_month = format(as.POSIXct(timestamp, origin="1970-1-1"), format="%Y%m")) %>%
  mutate(timestamp_date = format(as.POSIXct(timestamp, origin="1970-1-1"), format="%Y%m%d"))


# Validation set will be 10% of MovieLens data

set.seed(123)
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") %>%
  semi_join(edx, by = "timestamp_date")

# 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)

# End of the provided code
```

Data exploration and visualization

First, check the data structure and basic statistics. The target variable is rating, and a model that predicts this rating is created using other variables.

```r
# Data overview
# Check the first few lines
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 3      1     231      5 838983392          Dumb & Dumber (1994)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
```

```
## 6      1     329      5 838983392 Star Trek: Generations (1994)
##                             genres timestamp_year timestamp_month
## 1                   Comedy|Romance           1996          199608
## 2            Action|Crime|Thriller           1996          199608
## 3                           Comedy           1996          199608
## 4  Action|Drama|Sci-Fi|Thriller           1996          199608
## 5       Action|Adventure|Sci-Fi           1996          199608
## 6 Action|Adventure|Drama|Sci-Fi           1996          199608
##   timestamp_date
## 1       19960802
## 2       19960802
## 3       19960802
## 4       19960802
## 5       19960802
## 6       19960802
```

```r
# Check the data structure
str(edx)
```

```
## 'data.frame':    9000065 obs. of  9 variables:
##  $ userId         : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId        : num  122 185 231 292 316 329 355 356 362 364 ...
##  $ rating         : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp      : int  838985046 838983525 838983392 838983421 838983392 838983392 838984474 83898
##  $ title          : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb & Dumber (1994)" "Outbreak (1995
##  $ genres         : chr  "Comedy|Romance" "Action|Crime|Thriller" "Comedy" "Action|Drama|Sci-Fi|Thril
##  $ timestamp_year : chr  "1996" "1996" "1996" "1996" ...
##  $ timestamp_month: chr  "199608" "199608" "199608" "199608" ...
##  $ timestamp_date : chr  "19960802" "19960802" "19960802" "19960802" ...
```

```r
# Check basic statistics
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 :35741   Median : 1834   Median :4.000   Median :1.035e+09
##  Mean   :35873   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09
##  3rd Qu.:53612   3rd Qu.: 3624   3rd Qu.:4.000   3rd Qu.:1.127e+09
##  Max.   :71567   Max.   :65133   Max.   :5.000   Max.   :1.231e+09
##     title              genres          timestamp_year
##  Length:9000065     Length:9000065     Length:9000065
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##  timestamp_month    timestamp_date
##  Length:9000065     Length:9000065
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
```

The timestamp column is stored in UNIX time, and will have to be modified if it is to be incorporated into a predictive model. It should be noted that the genre column is stored with the delimiter character |.

Next, check the number of unique users, movies, and genre combinations.

```
# Unique count of movies, users and genres
n_distinct(edx$movieId)
```

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

```
n_distinct(edx$userId)
```

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

```
n_distinct (edx$genres)
```

```
## [1] 797
```

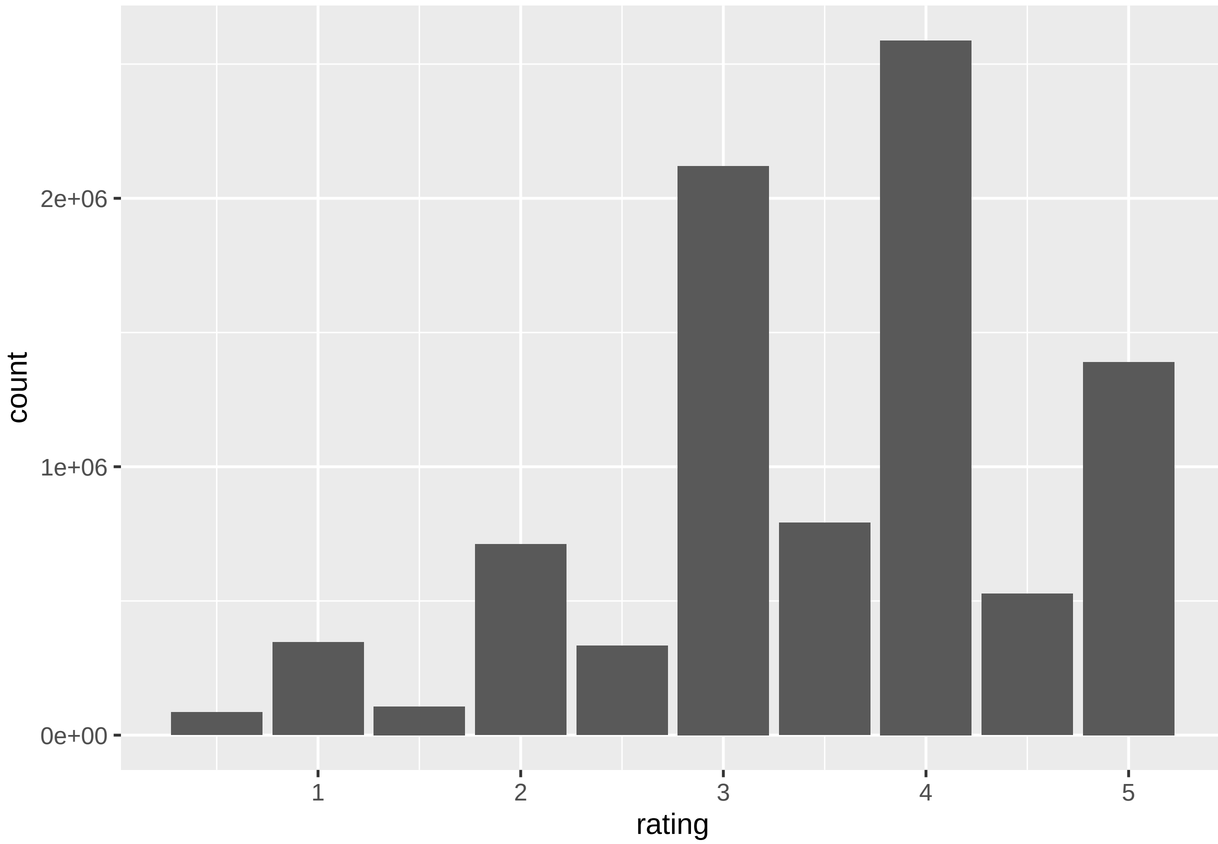
There are 10677 movie IDs. There are 69878 users. There are 797 genre combinations.

To see user ratings trends, check the distribution of rating that is the objective variable. First, plot the number of records for each rating value.

```
# Ratings count
edx %>% group_by(rating) %>% summarize(count=n()) %>%
  arrange(desc(rating))
```

```
## # A tibble: 10 x 2
##    rating   count
##     <dbl>   <int>
##  1    5   1390105
##  2    4.5  526745
##  3    4   2588490
##  4    3.5  791566
##  5    3   2120854
##  6    2.5  333122
##  7    2    711383
##  8    1.5  106577
##  9    1    345790
## 10    0.5   85433
```

```
# plot
edx %>% group_by(rating) %>%
  summarize(count = n()) %>%
  ggplot(aes(x = rating, y = count)) +
  geom_bar(stat = "identity")
```



The plot is not normally distributed. It can be seen that many users tend to rate with integer values rather than decimal values.

Model Building

Create test dataset and train dataset

The training data set was divided into two, and 10% was set to verify the accuracy of the model.

```
# set.seed
set.seed(123)
# 10% of the data is used as a test set to verify the accuracy of the model
test_ind <- createDataPartition(y = edx$rating, times = 1, p = .1, list=FALSE)
train_ds <- edx[-test_ind,]
test_ds_temp <- edx[test_ind,]

# Make sure userId and movieId in test dataset are also in train dataset
test_ds <- test_ds_temp %>%
  semi_join(train_ds, by = "movieId") %>%
  semi_join(train_ds, by = "userId") %>%
  semi_join(train_ds, by = "timestamp_date")

# Add rows removed from test_ds_temp dataset back into train dataset
rmvd <- anti_join(test_ds_temp, test_ds)
```

```
## Joining, by = c("userId", "movieId", "rating", "timestamp", "title", "genres", "timestamp_year", "ti
```

```r
train_ds <- rbind(train_ds, rmvd)
```

Create a simple prediction model with the average rating value as the predicted value. Based on this, build a model while adding variables to increase accuracy.

```r
# Model.1: Computing predicted ratings based on the average of ratings
mu_simple <- mean(train_ds$rating)
mu_simple
```

```
## [1] 3.512509
```

```r
# RMSE calculation
mod1_rmse <- RMSE(test_ds$rating, mu_simple)
mod1_rmse
```

```
## [1] 1.061372
```

```r
# RMSE of each model is stored in a table
rmse_table <- tibble(Model = "Model.1", Method="Average rating model",
                                RMSE = mod1_rmse)
rmse_table %>% knitr::kable()
```

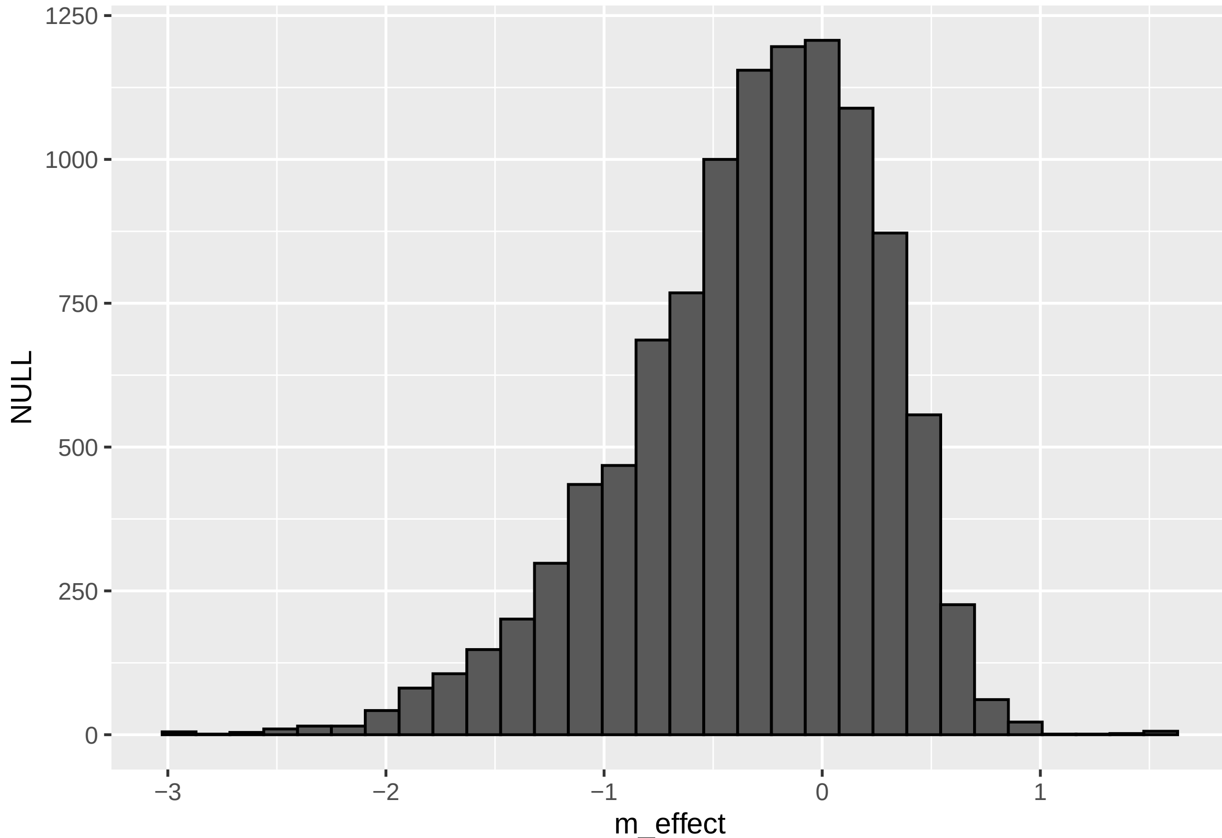
Model	Method	RMSE
Model.1	Average rating model	1.061372

The average rating in the training set is 3.512403. RMSE is as shown in the table.

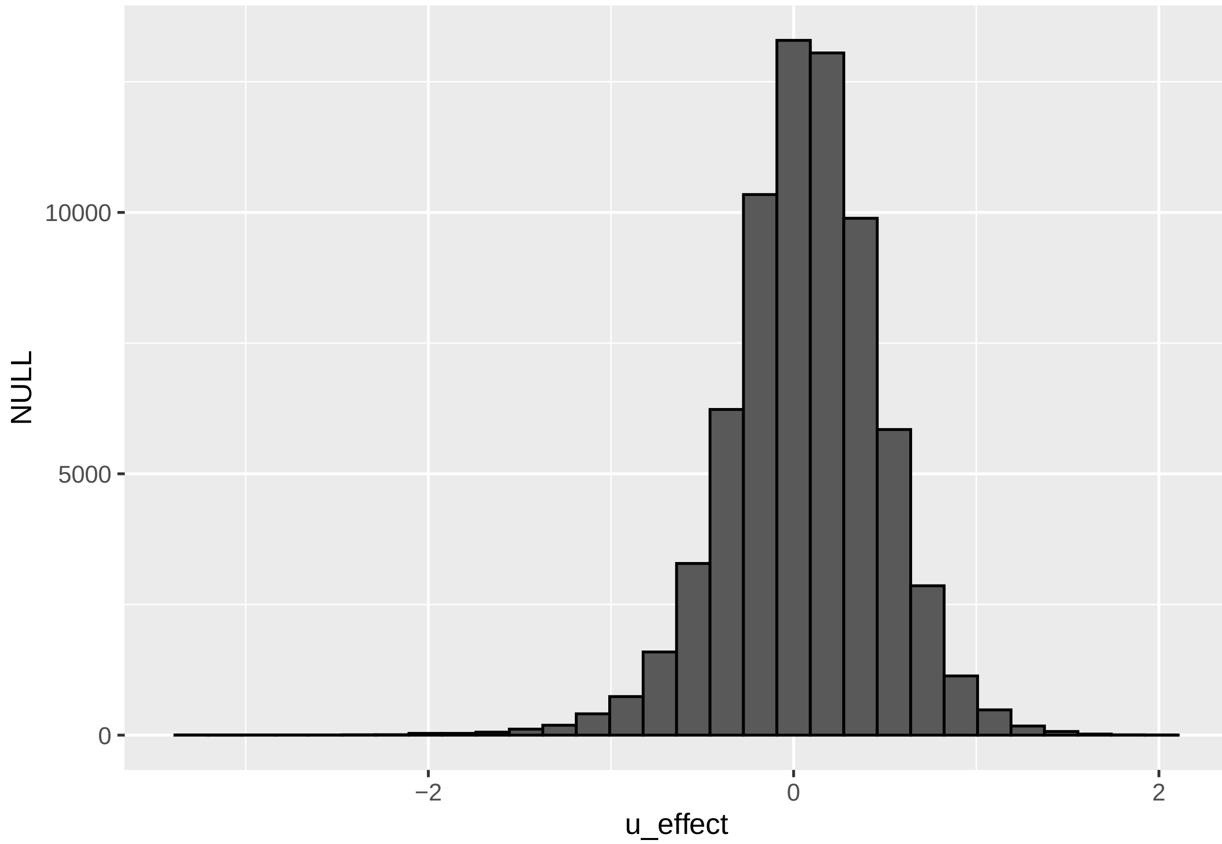
Next, create a model that incorporates the effects of movie ID and user ID.

```r
# Model.2: Computing predicted ratings based on movie effects and user effects
mu <- mean(train_ds$rating)

# Add movie effect
movie_effects <- train_ds %>%
  group_by(movieId) %>%
  summarize(m_effect = mean(rating - mu))
# Check the effect distribution of movie effect
movie_effects %>% qplot(m_effect, geom ="histogram", bins = 30, data = ., color = I("black"))
```




```r
# Add user effect
user_effects <- train_ds %>%
  left_join(movie_effects, by='movieId') %>%
  group_by(userId) %>%
  summarize(u_effect = mean(rating - mu - m_effect))
# Check the effect distribution of user effect
user_effects %>% qplot(u_effect, geom ="histogram", bins = 30, data = ., color = I("black"))
```




```
# Predict ratings
pred_ratings <- test_ds %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  mutate(pred = mu + m_effect + u_effect)
# RMSE calculation
mod2_rmse <- RMSE(test_ds$rating, pred_ratings$pred)
mod2_rmse
```

```
## [1] 0.8667407
```

```
# Check the maximum and minimum predicted values
max(pred_ratings$pred)
```

```
## [1] 6.088298
```

```
min(pred_ratings$pred)
```

```
## [1] -0.5208314
```

The maximum and minimum predicted values exceed the possible range of 0 to 5, respectively. A process of replacing a predicted value less than 0 with 0 and a predicted value exceeding 5 with 5 is performed.

```
# Adjusting the range of possible values (0~5)
pred_ratings$pred[pred_ratings$pred > 5] <- 5
max(pred_ratings$pred)
```

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

```r
pred_ratings$pred[pred_ratings$pred < 0] <- 0
min(pred_ratings$pred)
```

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

```r
# RMSE calculation
mod2_rmse <- RMSE(test_ds$rating, pred_ratings$pred)
mod2_rmse
```

```
## [1] 0.8665509
```

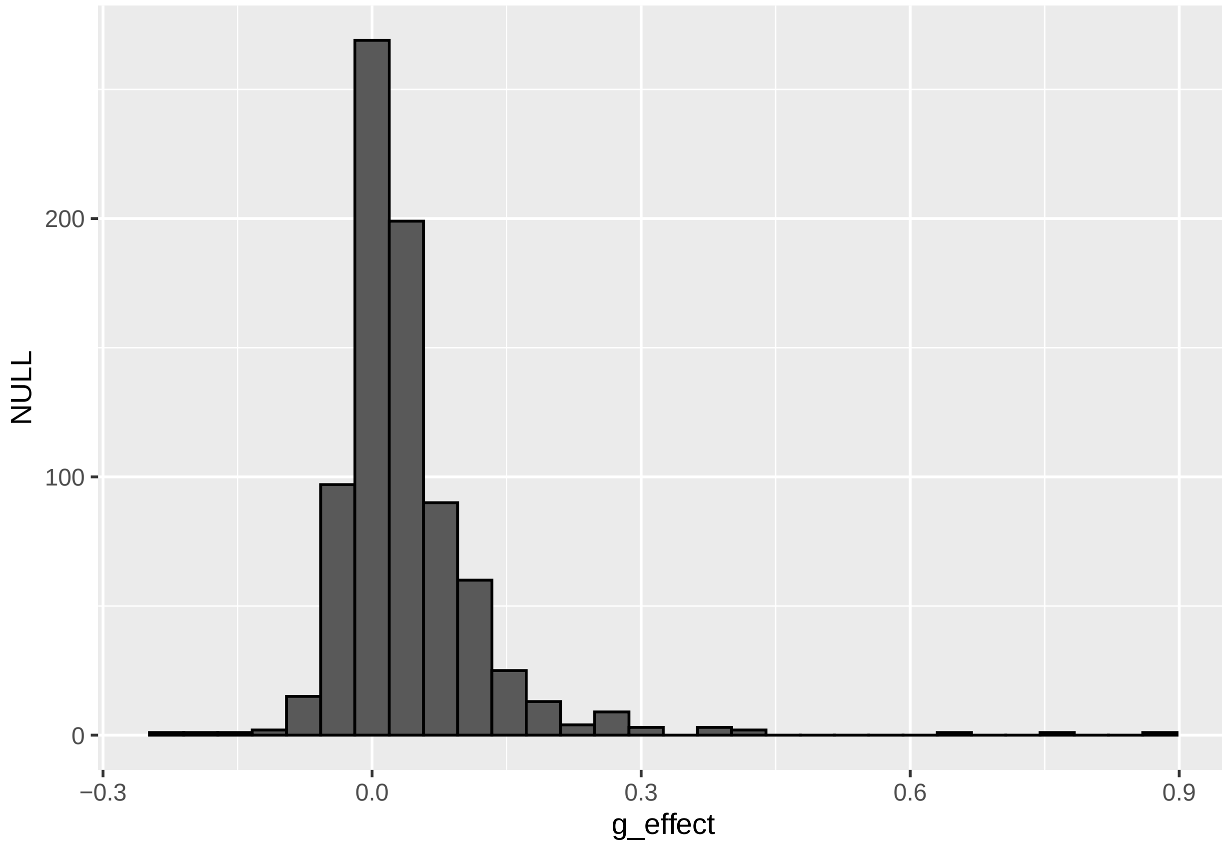
```r
# Store the RMSE of this model in a table
rmse_table <- bind_rows(rmse_table,
                        tibble(Model = "Model.2", Method="Movie & User Effect Model",
                                    RMSE = mod2_rmse ))
rmse_table %>% knitr::kable()
```

Model	Method	RMSE
Model.1	Average rating model	1.0613716
Model.2	Movie & User Effect Model	0.8665509

RMSE is as shown in the table. The accuracy is higher than the model using only the average value.

In addition, the impact of ratings on the combination of genres will be incorporated into the model.

```r
# Model.3: Computing predicted ratings based on movie, user & genre effects
# Add genre effect
genre_effects <- train_ds %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  group_by(genres) %>%
  summarize(g_effect = mean(rating - mu - m_effect - u_effect))
# Check the effect distribution of genre effect
genre_effects %>% qplot(g_effect, geom ="histogram", bins = 30, data = ., color = I("black"))
```




```r
# Predict ratings
pred_ratings <- test_ds %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  left_join(genre_effects, by='genres') %>%
  mutate(pred = mu + m_effect + u_effect + g_effect)
# RMSE calculation
mod3_rmse <- RMSE(test_ds$rating, pred_ratings$pred)
mod3_rmse
```

```
## [1] 0.8663226
```

```r
# Check the maximum and minimum predicted values
max(pred_ratings$pred)
```

```
## [1] 6.108714
```

```r
min(pred_ratings$pred)
```

```
## [1] -0.519349
```

As before, correct the predicted value outside the range of 0 to 5.

```r
# Adjusting the range of possible values (0~5)
pred_ratings$pred[pred_ratings$pred > 5] <- 5
max(pred_ratings$pred)
```

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

```r
pred_ratings$pred[pred_ratings$pred < 0] <- 0
min(pred_ratings$pred)
```

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

```r
# RMSE calculation
mod3_rmse <- RMSE(test_ds$rating, pred_ratings$pred)
# Store the RMSE of this model in a table
rmse_table <- bind_rows(rmse_table,
                        tibble(Model = "Model.3", Method="Movie & User & Genre Effect Model",
                               RMSE = mod3_rmse ))
rmse_table %>% knitr::kable()
```

Model	Method	RMSE
Model.1	Average rating model	1.0613716
Model.2	Movie & User Effect Model	0.8665509
Model.3	Movie & User & Genre Effect Model	0.8661249

Next, the effect of time on ratings is incorporated into the model. First, replace the date and time expressed in unixtime with the format "1970-1-1".

```r
# Convert timestamp to year (Sample code below)
unixtime = 1459995330
format(as.POSIXct(unixtime, origin="1970-1-1"), format="%Y%m")
```

```
## [1] "201604"
```

Using the above code, the UNIX time is converted to a notation in arbitrary units.

There are three types of time units to be incorporated into the model: year units, year / month units, and year / month / day units. First, the impact of the rating year on the rating is incorporated into the model.

```r
# Model.4_1: Computing predicted ratings based on movie, user, genre and year effects
# Add year effect
year_effects <- edx %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by= 'userId') %>%
  left_join(genre_effects, by='genres') %>%
  group_by(timestamp_year) %>%
  summarize(y_effect = mean(rating - mu - m_effect - u_effect -  g_effect))


# Predict ratings
pred_ratings <- test_ds %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by= 'userId') %>%
  left_join(genre_effects, by='genres') %>%
  left_join(year_effects, by= 'timestamp_year') %>%
  mutate(pred = mu + m_effect + u_effect + g_effect + y_effect)

# RMSE calculation
mod4_1_rmse <- RMSE(test_ds$rating, pred_ratings$pred)
mod4_1_rmse
```

```
## [1] 0.866315
```

```
# Adjusting the range of possible values (0~5)
pred_ratings$pred[pred_ratings$pred > 5] <- 5
pred_ratings$pred[pred_ratings$pred < 0] <- 0
# RMSE calculation
mod4_1_rmse <- RMSE(test_ds$rating, pred_ratings$pred)
mod4_1_rmse
```

```
## [1] 0.8661143
```

```
# Store the RMSE of this model in a table
rmse_table <- bind_rows(rmse_table,
                        tibble(Model = "Model.4_1", Method="year Effect Model",
                               RMSE = mod4_1_rmse ))
rmse_table %>% knitr::kable()
```

Model	Method	RMSE
Model.1	Average rating model	1.0613716
Model.2	Movie & User Effect Model	0.8665509
Model.3	Movie & User & Genre Effect Model	0.8661249
Model.4_1	year Effect Model	0.8661143

RMSE is as shown in the table. (Predictions are in the 0 to 5 range as before.)

Next, the impact of the rating year and month on the rating is incorporated into the model.

```
# Model.4_2: Computing predicted ratings based on movie, user, genre and month effects
# Add year effect
month_effects <- edx %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by= 'userId') %>%
  left_join(genre_effects, by='genres') %>%
  group_by(timestamp_month) %>%
  summarize(month_effect = mean(rating - mu - m_effect - u_effect - g_effect))


# Predict ratings
pred_ratings <- test_ds %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by= 'userId') %>%
  left_join(genre_effects, by='genres') %>%
  left_join(month_effects, by= 'timestamp_month') %>%
  mutate(pred = mu + m_effect + u_effect + g_effect + month_effect)

# RMSE calculation
mod4_2_rmse <- RMSE(test_ds$rating, pred_ratings$pred)
mod4_2_rmse
```

```
## [1] 0.8662827
```

```
# Adjusting the range of possible values (0~5)
pred_ratings$pred[pred_ratings$pred > 5] <- 5
pred_ratings$pred[pred_ratings$pred < 0] <- 0

# RMSE calculation
```

```r
mod4_2_rmse <- RMSE(test_ds$rating, pred_ratings$pred)
mod4_2_rmse
```

```
## [1] 0.8660809
```

```r
# Store the RMSE of this model in a table
rmse_table <- bind_rows(rmse_table,
                        tibble(Model = "Model.4_2", Method="month Effect Model",
                               RMSE = mod4_2_rmse ))
rmse_table %>% knitr::kable()
```

Model	Method	RMSE
Model.1	Average rating model	1.0613716
Model.2	Movie & User Effect Model	0.8665509
Model.3	Movie & User & Genre Effect Model	0.8661249
Model.4_1	year Effect Model	0.8661143
Model.4_2	month Effect Model	0.8660809

RMSE is as shown in the table. (Predictions are in the 0 to 5 range as before.)

Finally, the impact of the rating date on the rating is incorporated into the model.

```r
# Model.4_3: Computing predicted ratings based on movie, user, genre and day effects
# Add year effect
date_effects <- edx %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by= 'userId') %>%
  left_join(genre_effects, by='genres') %>%
  group_by(timestamp_date) %>%
  summarize(d_effect = mean(rating - mu - m_effect - u_effect -  g_effect))


# Predict ratings
pred_ratings <- test_ds %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by= 'userId') %>%
  left_join(genre_effects, by='genres') %>%
  mutate(timestamp_year = format(as.POSIXct(timestamp, origin="1970-1-1"), format="%Y%m%d")) %>%
  left_join(date_effects, by= 'timestamp_date') %>%
  mutate(pred = mu + m_effect + u_effect + g_effect + d_effect)

# RMSE calculation
mod4_3_rmse <- RMSE(test_ds$rating, pred_ratings$pred)
mod4_3_rmse
```

```
## [1] 0.8653333
```

```r
# Adjusting the range of possible values (0~5)
pred_ratings$pred[pred_ratings$pred > 5] <- 5
pred_ratings$pred[pred_ratings$pred < 0] <- 0
# RMSE calculation
mod4_3_rmse <- RMSE(test_ds$rating, pred_ratings$pred)
mod4_3_rmse
```

```
## [1] 0.86513
```

```r
# Store the RMSE of this model in a table
rmse_table <- bind_rows(rmse_table,
                        tibble(Model = "Model.4_3", Method="date Effect Model",
                               RMSE = mod4_3_rmse ))
rmse_table %>% knitr::kable()
```

Model	Method	RMSE
Model.1	Average rating model	1.0613716
Model.2	Movie & User Effect Model	0.8665509
Model.3	Movie & User & Genre Effect Model	0.8661249
Model.4_1	year Effect Model	0.8661143
Model.4_2	month Effect Model	0.8660809
Model.4_3	date Effect Model	0.8651300

RMSE is as shown in the table. (Predictions are in the 0 to 5 range as before.)

(Although this model.4_3 does not reach the target RMSE value of 0.8649) Since the last model had the best RMSE with the test set, this model is applied to the validation set for final evaluation.

Testing the final model on the Validation dataset

```r
# Testing the final model on the Validation dataset
predicted_ratings <- validation %>%
  left_join(movie_effects, by='movieId') %>%
  left_join(user_effects, by='userId') %>%
  left_join(genre_effects, by= 'genres') %>%
  left_join(date_effects, by= 'timestamp_date') %>%
  mutate(pred = mu + m_effect + u_effect + g_effect + d_effect)

head(predicted_ratings)
```

```
##   userId movieId rating  timestamp
## 1      1     480    5.0  838983653
## 2      1     539    5.0  838984068
## 3      1     586    5.0  838984068
## 4      2     260    5.0  868244562
## 5      2    1544    3.0  868245920
## 6      3     590    3.5 1136075494
##                                                         title
## 1                                        Jurassic Park (1993)
## 2                                 Sleepless in Seattle (1993)
## 3                                           Home Alone (1990)
## 4 Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)
## 5      Lost World: Jurassic Park, The (Jurassic Park 2) (1997)
## 6                                   Dances with Wolves (1990)
##                                   genres timestamp_year timestamp_month
## 1        Action|Adventure|Sci-Fi|Thriller           1996          199608
## 2                    Comedy|Drama|Romance           1996          199608
## 3                         Children|Comedy           1996          199608
## 4                 Action|Adventure|Sci-Fi           1997          199707
## 5 Action|Adventure|Horror|Sci-Fi|Thriller           1997          199707
## 6                 Adventure|Drama|Western           2006          200601
```

```
##   timestamp_date    m_effect   u_effect    g_effect     d_effect     pred
## 1       19960802  0.15127414  1.7538035 -0.02320686  0.025006273 5.419386
## 2       19960802  0.02607313  1.7538035  0.00285945  0.025006273 5.320251
## 3       19960802 -0.45690925  1.7538035 -0.02231849  0.025006273 4.812091
## 4       19970707  0.71052273 -0.4269085 -0.02354021 -0.004033319 3.768550
## 5       19970707 -0.55950615 -0.4269085 -0.02041403 -0.004033319 2.501647
## 6       20060101  0.22846480  0.3076306 -0.04735892  0.025781405 4.027027
```

Replace the value so that the predicted value is in the range of 0 to 5.

```r
# Adjusting the range of possible values (0~5)
 predicted_ratings$pred[predicted_ratings$pred > 5] <- 5
 max(predicted_ratings$pred)
```

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

```r
 predicted_ratings$pred[predicted_ratings$pred < 0] <- 0
 min(predicted_ratings$pred)
```

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

Computing final RMSE

```r
# final RMSE calculation
model_rmse <- RMSE(predicted_ratings$pred, validation$rating)
model_rmse
```

```
## [1] 0.8644318
```

```r
# Store the final RMSE of this model in a table
rmse_table <- bind_rows(rmse_table,
                        tibble(Model = "*final RMSE", Method="Model.4_3 year Effect Model",
                                     RMSE = model_rmse ))
rmse_table %>% knitr::kable()
```

Model	Method	RMSE
Model.1	Average rating model	1.0613716
Model.2	Movie & User Effect Model	0.8665509
Model.3	Movie & User & Genre Effect Model	0.8661249
Model.4_1	year Effect Model	0.8661143
Model.4_2	month Effect Model	0.8660809
Model.4_3	date Effect Model	0.8651300
final RMSE	Model.4_3 year Effect Model	0.8644318

The RMSE in the validation set using the final selected model is as shown in the table. The target RMSE value of 0.8649 was achieved.

conclusion

It was found that only a simple model gives a certain level of accuracy. Making a simple model with accuracy is also great in terms of accountability. Since the influence of each variable on the predicted value is shown as a clear value, it will be easy to use for explanation to the decision maker. I also tried a machine learning model such as Random Forest, but a memory out error occurred and the model was not built. The memory problem may be solved by dividing the data set or selecting variables, but that is left for future work.