Movie Rating Prediction-Harvard X Data Science

MovieLens Capstone Project

Content * Part 1 Introduction * Part 2 Data Download & Wrangling * Part 3 Exploratory Analysis * Part 4 Data Modeling * Part 5 Model Validation * Part 6 Conclusion

Part 1 Introduction

This MovieLens Dataset is originally from GroupLens.GroupLens Research has collected and made available rating data sets from the MovieLens web site (http://movielens.org (http://movielens.org)).They provide movie datasets with 100k, 1M, 10M, 20M data respectively. For this study, the 10M dataset is chosen here.

This dataset will be splited into two dataset: edx (90%) and validation (10%). The edx dataset will be used for training and testing model. The validation dataset will only be used for validation in Part 5.

To summerize, the steps for this project are: 1. Download the data 2. Glimpse the data and conduct data wrangling 3. Conduct exploratory analysis to further study the dataset. 4. Build linear model for the dataset and test the RMSE result. 5. Regularization and test the RMSE result.
6. Validate the linear model and find out the RMSE result.

After all above steps, the final RMSE result of the model is 0.8399034.

Part 2 Data Download & Wrangling

Download the data from movielens website

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-proje
ct.org")
```

```
## Loading required package: tidyverse
```

```
## ── Attaching packages ───────────────────────────────────────────
────────────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.0     ✓ purrr   0.3.4
## ✓ tibble  3.0.1     ✓ dplyr   0.8.5
## ✓ tidyr   1.0.2     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.5.0
```

```
## ── Conflicts ─────────────────────────────────────────────────────
──────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-pro
ject.org")
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
# 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))[movi
eId],
                                           title = as.character(title),
                                           genres = as.character(genres))

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

```
## 'data.frame':    10000054 obs. of  6 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 838
984474 838983653 838984885 838983707 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb & Dumber (1994)" "Ou
tbreak (1995)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Comedy" "Action|Drama
|Sci-Fi|Thriller" ...
```

```
# Validation set will be 10% of MovieLens data
set.seed(1)
# if using R 3.5 or earlier, use `set.seed(1)` instead
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FA
LSE)
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)
```

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

```
edx <- rbind(edx, removed)

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

The validation dataset will be used in the last step to validate the model only. In the following part, only the edx will be further studied and explored.

Explore edx 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 838
983653 838984885 838983707 838984596 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Outbreak (1995)" "Stargat
e (1994)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller" "Action|Drama|Sci-Fi|T
hriller" "Action|Adventure|Sci-Fi" ...
```

There are 9000055 observations and 6 variables in the edx dataset. We can also notice that the time is in the format of "timestamp", the title variable contains both the movie name and the year when the movie was released to the public, and the genre for each movie is a combination of a few general genres.

```
edx[!complete.cases(edx),]
```

```
0 rows
```

Since the returned results is 0 row, then we can know that there’s no missing value in this dataset.

Change the time format

```
library(stringi)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year
```

```
## The following objects are masked from 'package:dplyr':
##
##     intersect, setdiff, union
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
release <- stringi::stri_extract_last_regex(edx$title, "(\\d{4})", comments = TRUE )
%>% as.numeric()
edxtidy <- edx %>% mutate (Year_Released = release, Year_Rated = year(as_datetime(tim
estamp)), Month_Rated = month(as_datetime(timestamp)))%>%select(-timestamp)
head(edxtidy)
```

title <chr>	genres <chr>	Year_Released <dbl>	Ye
Boomerang (1992)	Comedy|Romance	1992	
Net, The (1995)	Action|Crime|Thriller	1995	
Outbreak (1995)	Action|Drama|Sci-Fi|Thriller	1995	
Stargate (1994)	Action|Adventure|Sci-Fi	1994	

Star Trek: Generations (1994)	Action|Adventure|Drama|Sci-Fi	1994
Flintstones, The (1994)	Children|Comedy|Fantasy	1994

6 rows | 5-9 of 9 columns

The timestamp in edx dataset has been changed and splited into two variables: Year_rated and Month_rated, so as to analyze whether there's seasonal trend or bias in rating time in later analysis.

The year when the movie was released to the public is also a part of movie title, therefore, the released years are extracted into a new variable: Year_Released.

Find out how many unique values in each variables.

```
number_user <- n_distinct(edxtidy$userId)
number_movie <- n_distinct(edxtidy$movieId)
number_genre <- n_distinct(edxtidy$genres)
number_Year_Released <- n_distinct(edxtidy$Year_Released)
number_Year_Rated <- n_distinct(edxtidy$Year_Rated)
number_Month_Rated <- n_distinct(edxtidy$Month_Rated)
number_glimpse <- data.table(edx_variable = c("User","Movie","Genre","Year_Release","
Year_Rated","Month_Rated"), variable_number = c(number_user,number_movie,number_genre
,number_Year_Released,number_Year_Rated,number_Month_Rated))
number_glimpse
```

edx_variable <chr>	variable_number <int>
User	69878
Movie	10677
Genre	797
Year_Release	94
Year_Rated	15
Month_Rated	12

6 rows

To sum up, in the edx dataset, there are 69878 users who rated total 10677 movies in 797 genres within 15 years. The years when the 10677 movies were first released to the public, haven been across 94 years during the past.

Part 3 Exploratory Analysis

Data Distritubion

Summarize Rating

```
summary(edxtidy$rating)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.500   3.000   4.000   3.512   4.000   5.000
```

```
edxtidy %>% group_by(rating) %>% summarize( n = n())
```

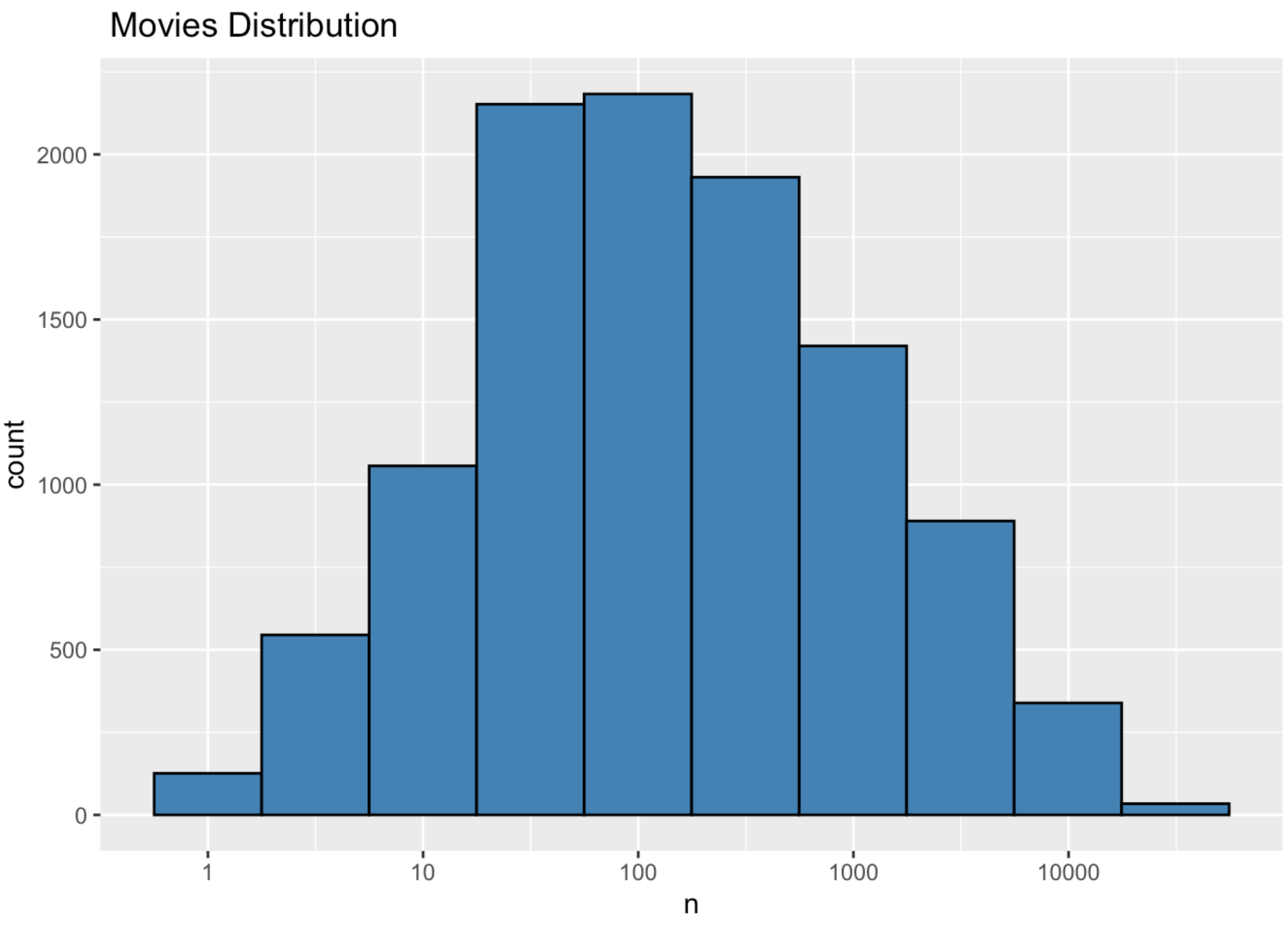
rating <dbl>	n <int>
0.5	85374
1.0	345679
1.5	106426
2.0	711422
2.5	333010
3.0	2121240
3.5	791624
4.0	2588430
4.5	526736
5.0	1390114

1-10 of 10 rows

The scale of rating starts from 0.5 to 5.0. Majority movies have been rated equal to or higher than 3.0.

Distribution of Movie Ratings

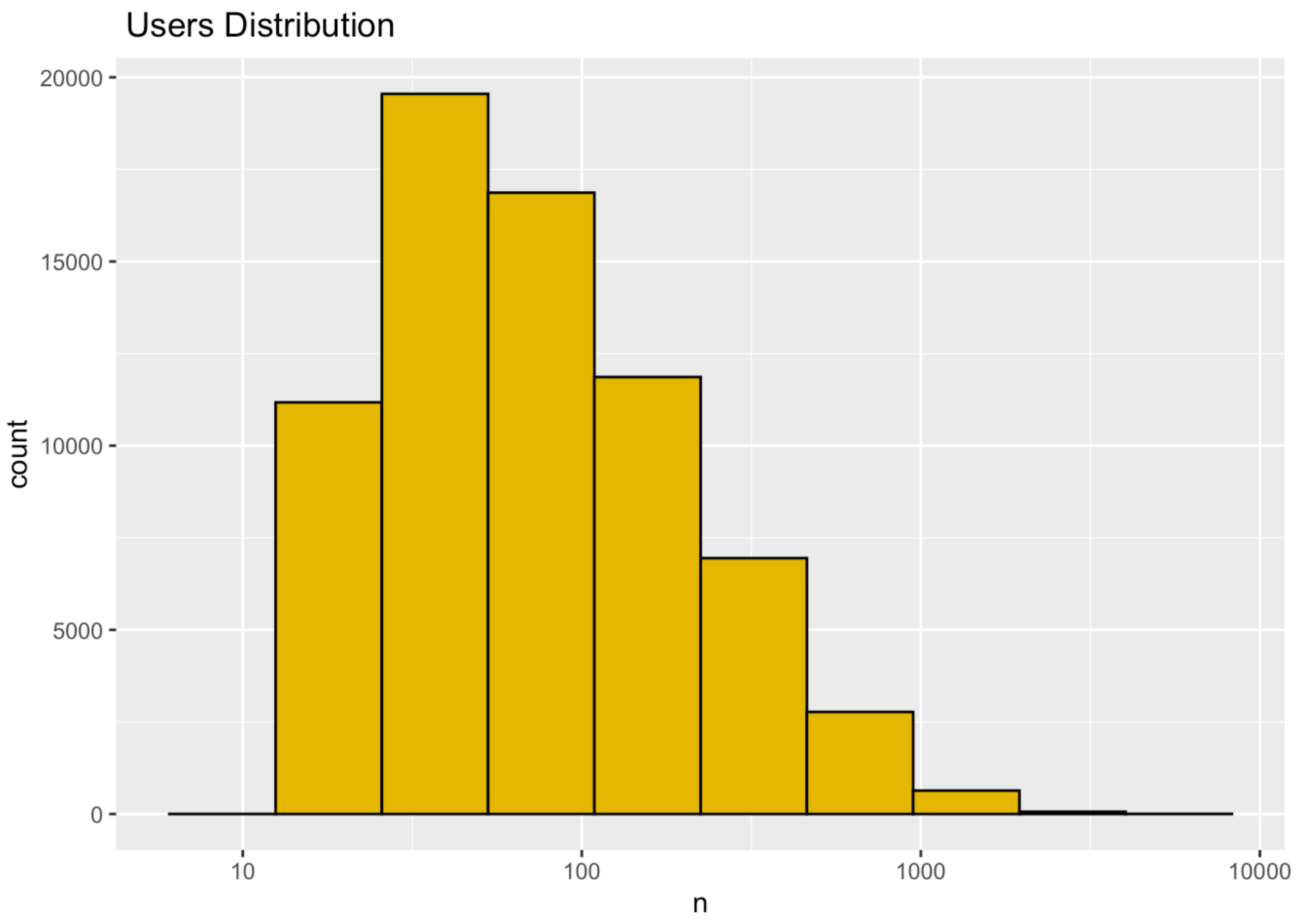
```
edxtidy %>% group_by(movieId) %>% summarize(n = n()) %>%
  ggplot(aes(n)) + geom_histogram(fill = "steelblue", color = "black", bins = 10) +
  scale_x_log10() +
  ggtitle(" Movies Distribution")
```



From above histogram, we can learn that, movie variable is a little right skewed.

Distribution of User Ratings

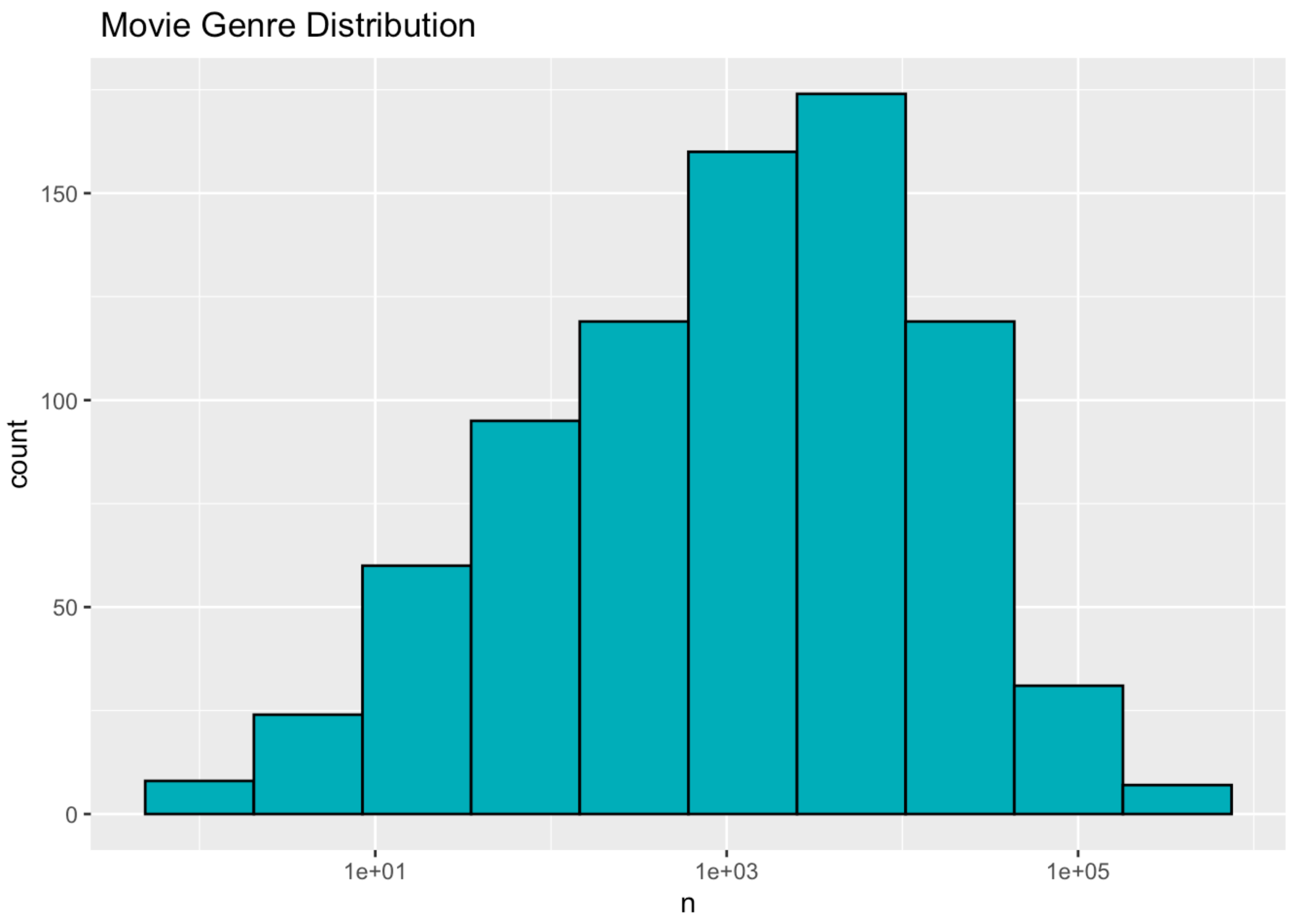
```
edxtidy %>% group_by(userId) %>% summarize(n = n()) %>%
  ggplot(aes(n)) + geom_histogram(fill = "#E7B800", color = "black", bins = 10) +
  scale_x_log10() +
  ggtitle(" Users Distribution")
```



From above histogram, we can learn that, user variable is obviously right skewed.

Distribution of Movie Genre

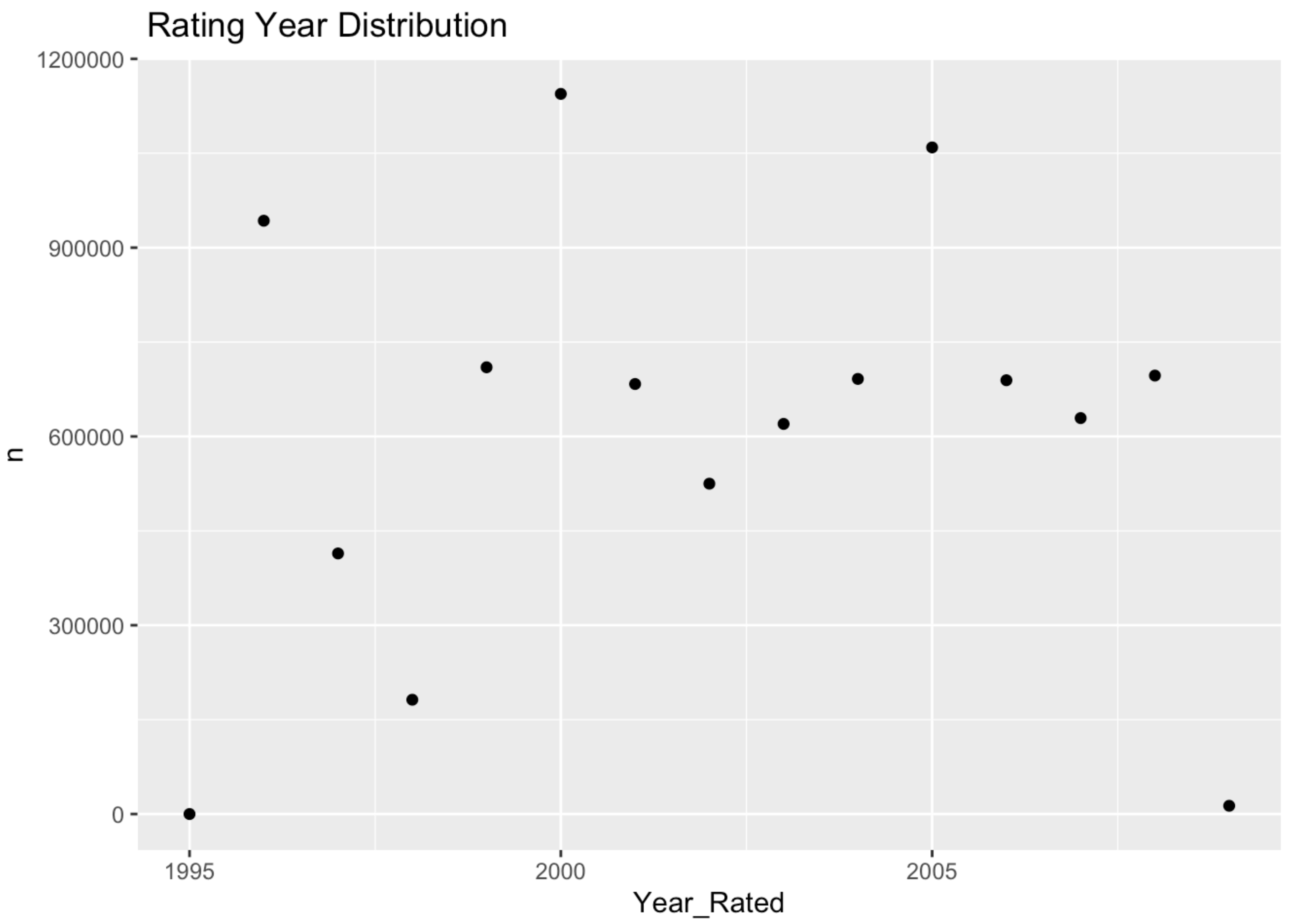
```
edxtidy %>% group_by(genres) %>% summarize(n = n()) %>%
  ggplot(aes(n)) + geom_histogram(fill = "#00AFBB", color = "black", bins = 10) +
  scale_x_log10() +
  ggtitle(" Movie Genre Distribution")
```



From above histogram, we can learn that, movie genre variable is left skewed.

Distribution of rating year

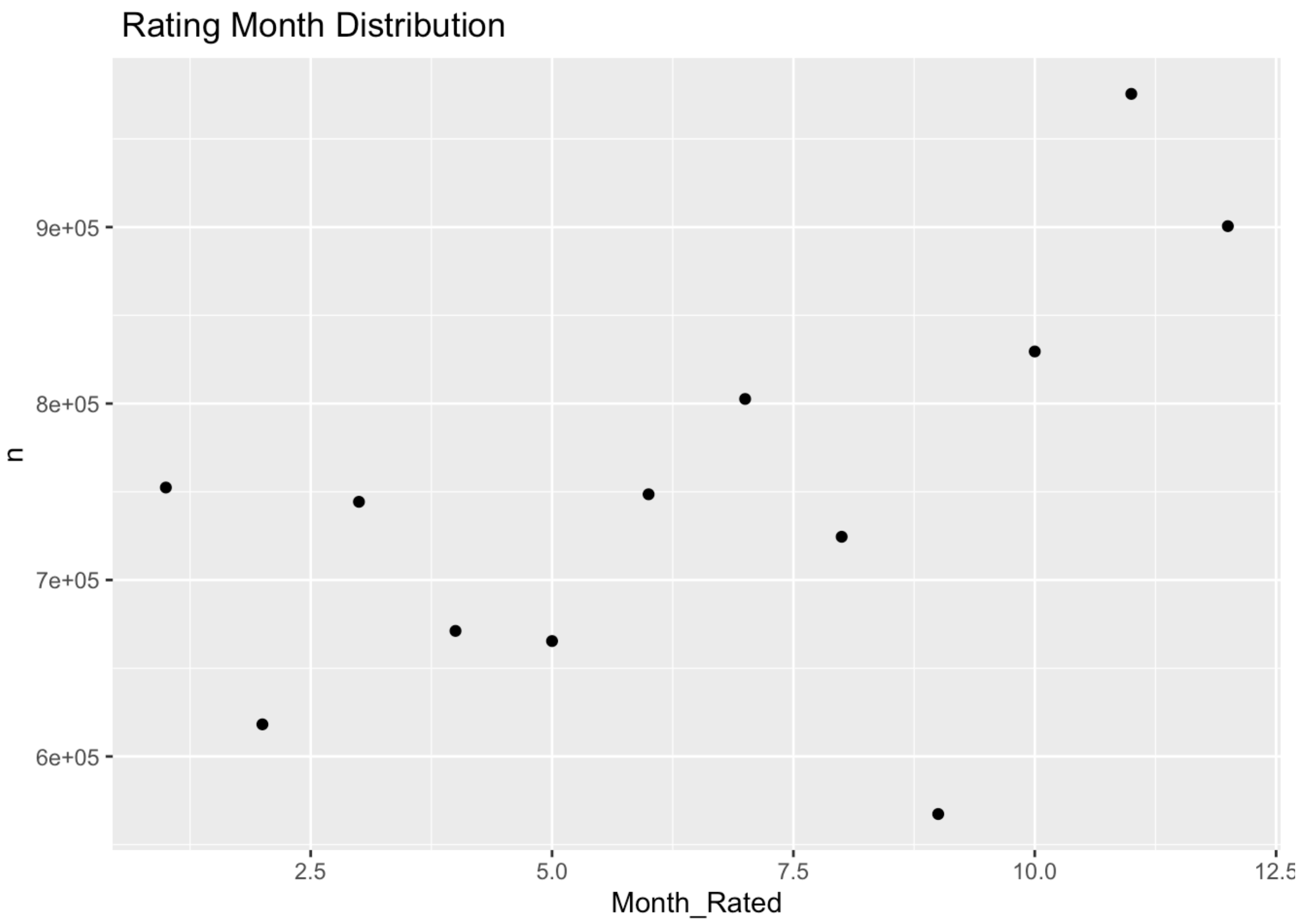
```
edxtidy %>% group_by(Year_Rated) %>% summarize(n = n())  %>%
  ggplot(aes(x = Year_Rated, y=n)) + geom_point() +
  ggtitle(" Rating Year Distribution")
```



From above chart, we can learn that, besides the first year 1995 and the last year, there have been plenty of ratings for all other years.

Distribution of rating month

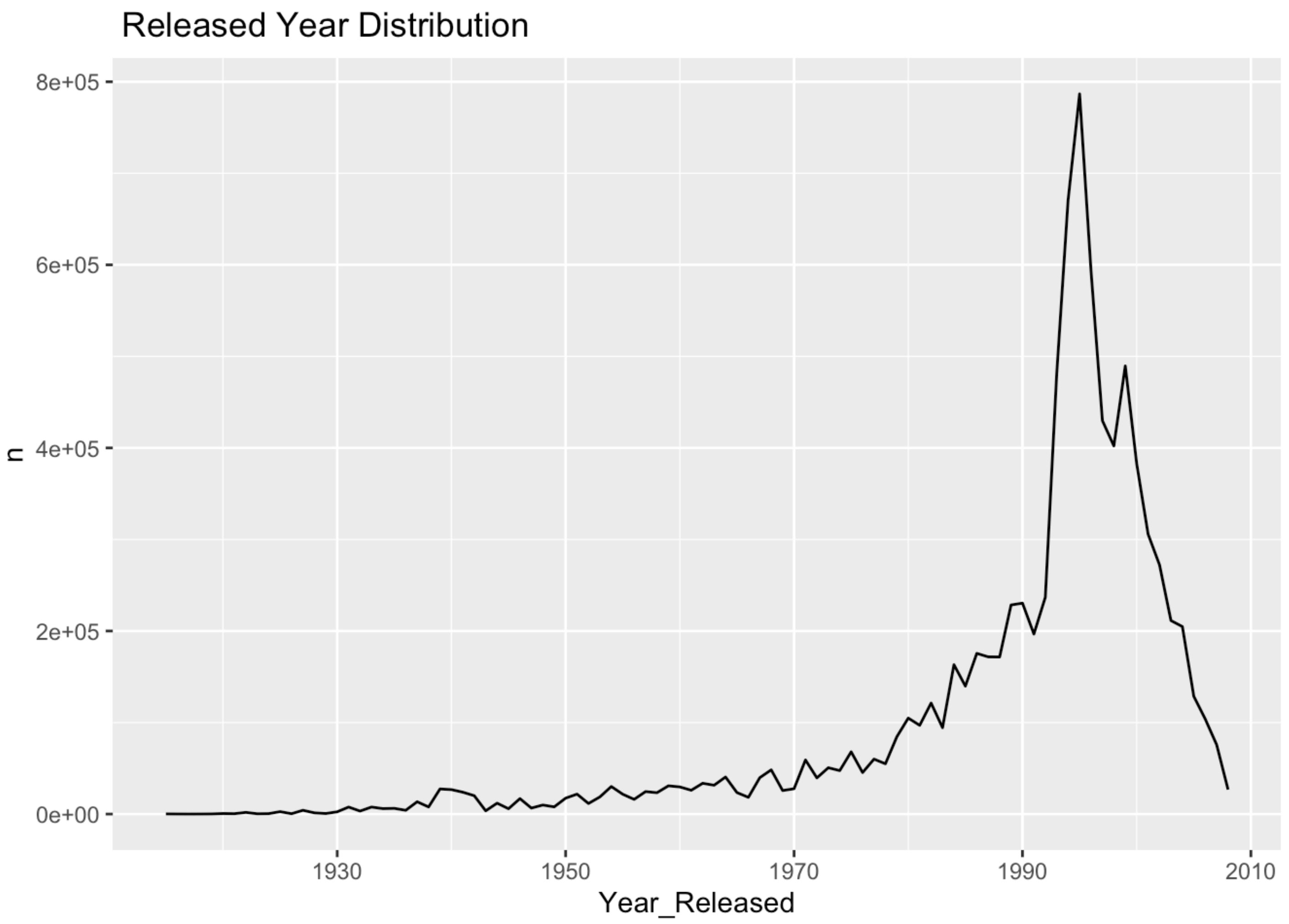
```
edxtidy %>% group_by(Month_Rated) %>% summarize(n = n()) %>%
  ggplot(aes(x = Month_Rated, y=n)) + geom_point() +
  ggtitle(" Rating Month Distribution")
```



From above chart, we can learn that, there are more ratings in the winter season October, November and December,maybe because of the Thanksgiving and X'mas holidays. Though there are variances in the number of rating month, the differences are not that severly.

Distribution of release year

```
edxtidy %>% group_by(Year_Released) %>% summarize(n = n())  %>%
  ggplot(aes(x = Year_Released, y=n)) + geom_line() +
  ggtitle(" Released Year Distribution")
```



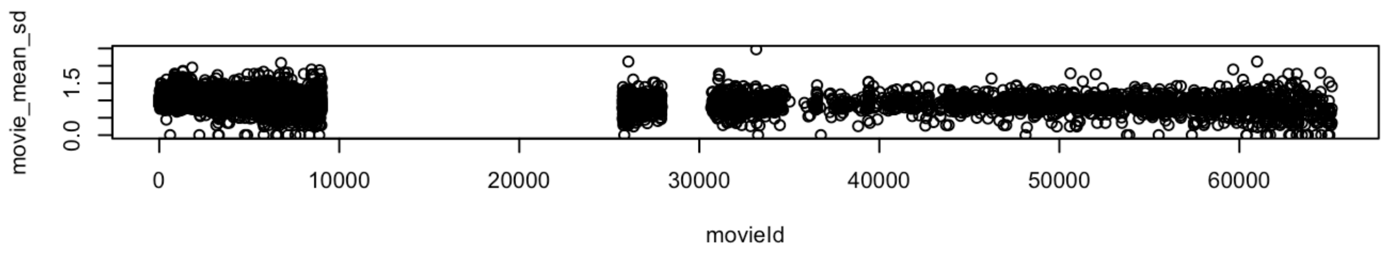
From above chart, we can learn that released year is sharply left skewed.

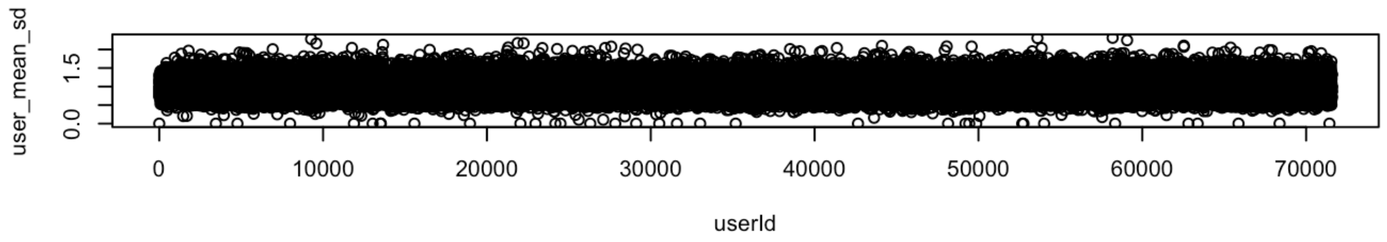
After study the distribution of all variables, Year_Released will tell more information than both Year_Rated and Month_Rated, therefore, for the following further exploration, the time when the user rated the movies won't be considered.

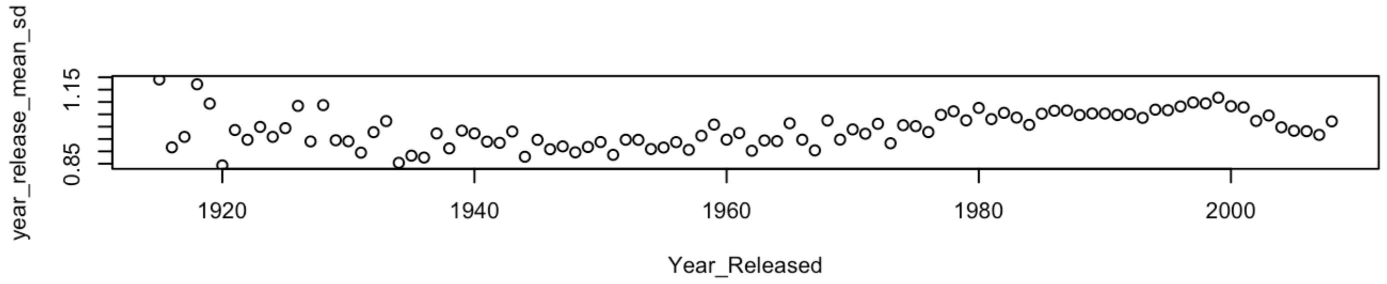
Futher Exploration

Explore the standard deviation of rating by Variables

```
par(mfrow=c(3,1))
edxtidy %>% group_by(movieId) %>% summarize(movie_mean_sd = sd(rating))%>%as.data.fra
me()%>%plot()
edxtidy %>% group_by(userId) %>% summarize(user_mean_sd = sd(rating))%>%as.data.frame
()%>%plot()
edxtidy %>% group_by(Year_Released) %>% summarize(year_release_mean_sd = sd(rating))%
>%as.data.frame()%>%plot()
```







From above chart, we can learn that, how the rating by each variable is spread out. And we can also tell that there are bias in movie preference, user personal preference and years released.

It is also noticeable that in the edx dataset, the genre in each observation is a combination. Before further exploration, genre variables needs to be tidied.

```
singlegenre <- edxtidy %>% separate_rows(genres, sep ="\\|")
head(singlegenre)
```

	userId <int>	movieId <dbl>	rating <dbl>	title <chr>	genres <chr>	Year_Released <dbl>	Year_Rated <dbl>	Month_Ra <dl
1	1	122	5	Boomerang (1992)	Comedy	1992	1996	
2	1	122	5	Boomerang (1992)	Romance	1992	1996	
3	1	185	5	Net, The (1995)	Action	1995	1996	
4	1	185	5	Net, The (1995)	Crime	1995	1996	
5	1	185	5	Net, The (1995)	Thriller	1995	1996	

```
6     1     292     5  Outbreak (1995)     Action              1995          1996
6 rows
```

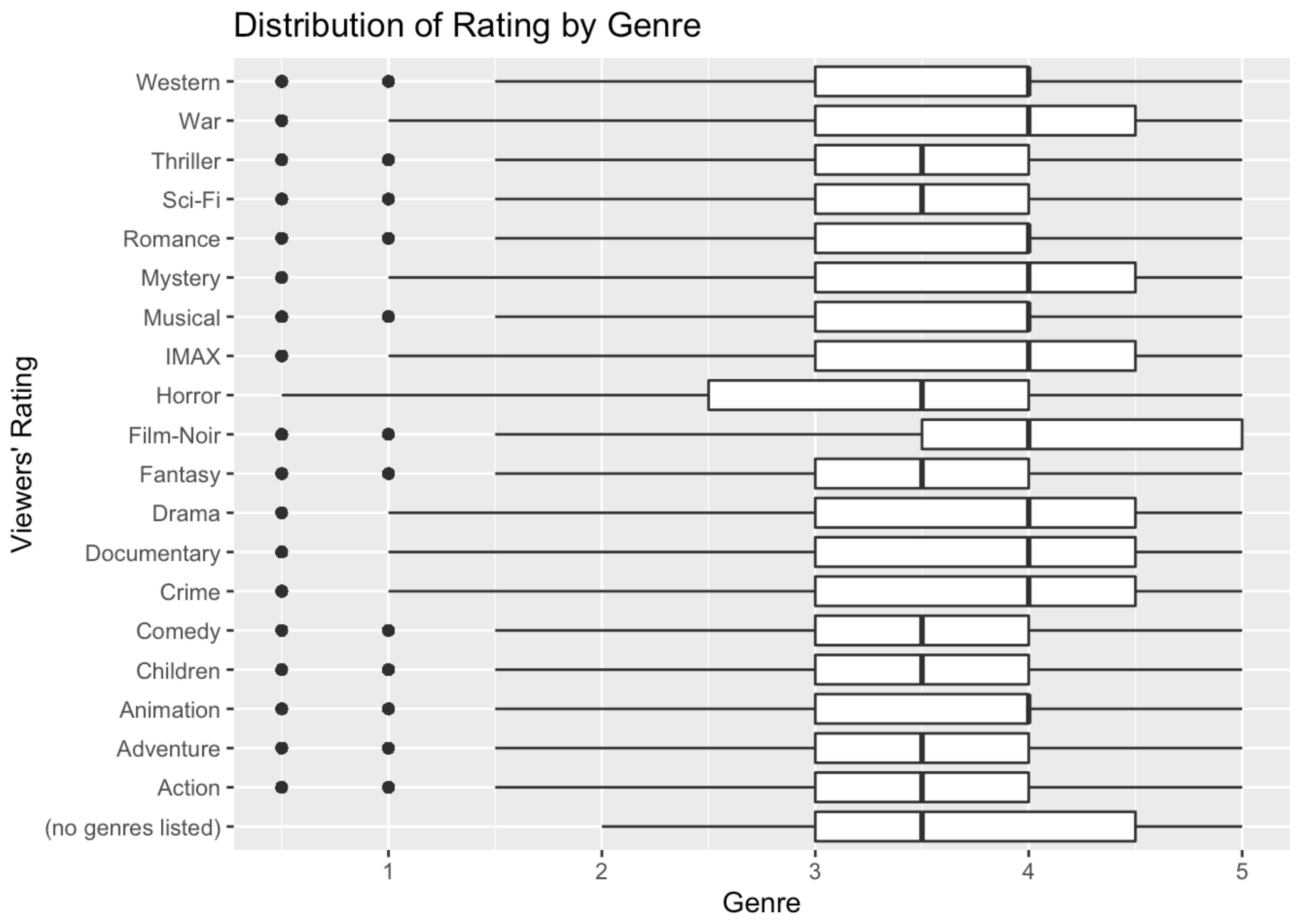
```
genreclass <- singlegenre%>%group_by(genres)%>%summarize(n = n())
genreclass
```

genres <chr>	**n** <int>
(no genres listed)	7
Action	2560545
Adventure	1908892
Animation	467168
Children	737994
Comedy	3540930
Crime	1327715
Documentary	93066
Drama	3910127
Fantasy	925637

1-10 of 20 rows Previous **1** 2 Next

There are 20 unique genres in total for all movies. Many movies have been assigned with mutiple genres combination.

```
library(ggplot2)
avg_genreclass <- singlegenre%>%group_by(genres)%>%summarize(avg_rate_genreclass = me
an(rating))
singlegenre %>%
  ggplot(aes(genres,rating)) + geom_boxplot()+ coord_flip()+
  labs(y = "Genre", x = "Viewers' Rating") +
  ggtitle("Distribution of Rating by Genre")
```

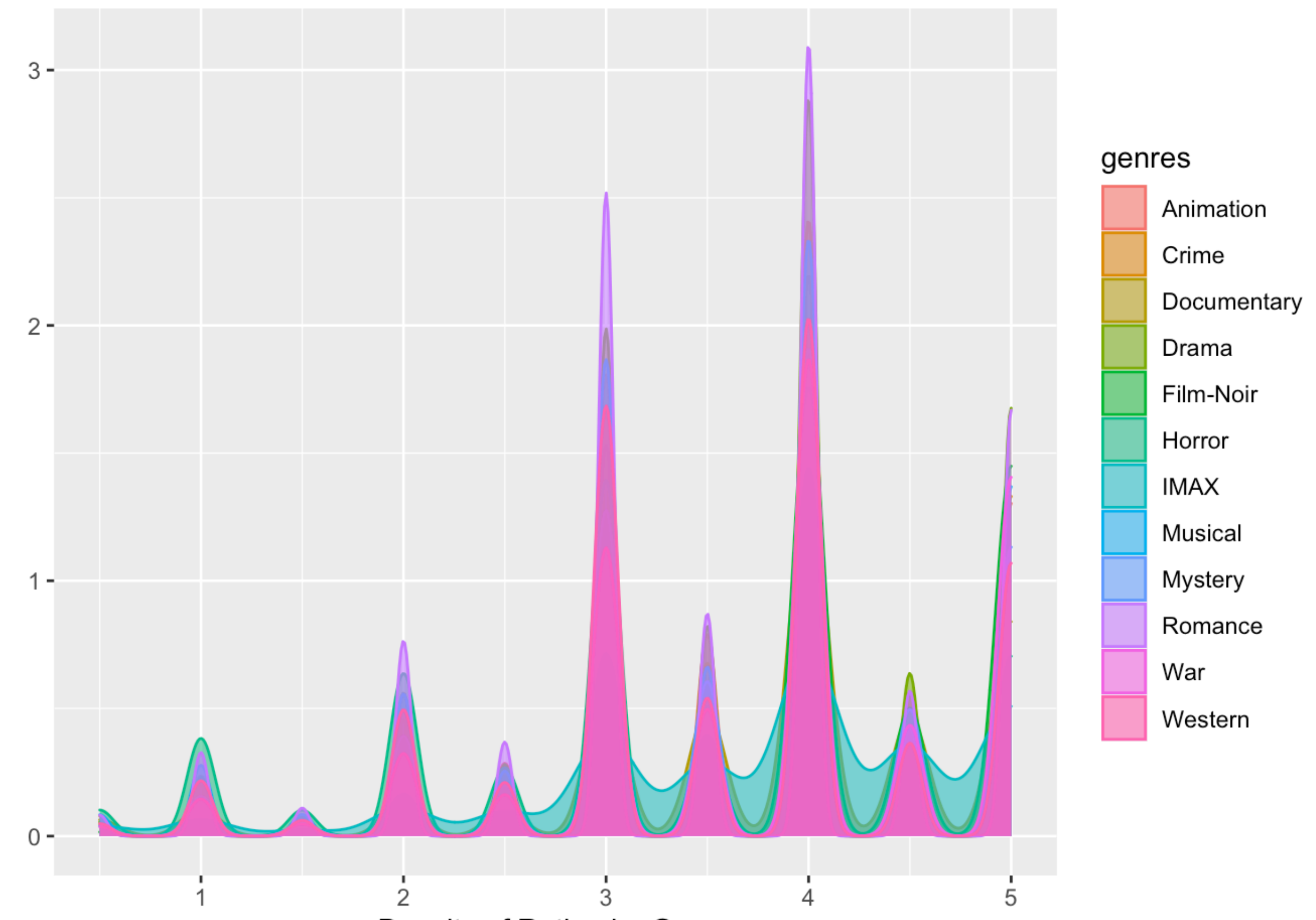


We can see from the boxplot that the Thriller, Sci-Fi, Fantasy, Comedy, Children, Adventure, Action are close to normally distributed with the medium close to the mean 3.5. Because only total 7 movies are in the category - (no genres listed), they can be ignored.

Besides the movie genres mentioned in previous paragraph, other genres are skewed.

Study more further in the disbrition of the skewed genre.

```
Skewedgenres <-c("Western","War","Romance","Mystery","Musical","IMAX","Horror","Film-
Noir","Drama","Documentary","Crime","Animation")
skewed_genre <- singlegenre%>%filter(genres %in% Skewedgenres )
skewed_genre %>%ggplot(aes(x = rating,color=genres,fill = genres))+
  geom_density(alpha=0.6) +
  ylab("") +
  xlab("Density of Rating by Genre")
```



Density of Rating by Genre

From above density chart, we can learn that ratings for these movie genres are generaly higher. There is rating bias for movie genre.

Part 4 Data Modeling

Split the edx dataset into train and test,use 20% of the dataset for testing purposes.

```
set.seed(1)
test_index2 <- createDataPartition(y = edxtidy$rating, times = 1, p = 0.2, list = FAL
SE)
train <- edxtidy[-test_index2,]
test <- edxtidy[test_index2,]
# Semi_join, so as to make sure the movies in the training set are also in test set.
test <- test %>% semi_join(train, by = "movieId") %>%semi_join(train, by = "userId")
nrow(train)
```

```
## [1] 7200043
```

```
nrow(test)
```

```
## [1] 1799966
```

There are 720043 observations for training, and 1799966 observations used for testing later.

Build Linear Model and test the performance using RSME.

Step 1 Start from the linear model assuming the same rating for all ratings with all the differences explained by random variation: predicted rate = true rate + epsilon. The least squares estimate of true rate, is the average rating of all movies across all users, and epsilon represents independent errors sampled from the same distribution centered at zero, then:

```
avg_rate_mean <- mean(train$rating)
avg_rate_mean
```

```
## [1] 3.512482
```

```
step1_rsme <- RMSE(test$rating,avg_rate_mean)
step1_rsme
```

```
## [1] 1.059904
```

The RSME for step 1 is 1.059904 which is too high.

Step 2 Add Movie Bias to the model

```
bias_movie_train <- train%>%group_by(movieId) %>% summarize(movie_bias_rate = mean(ra
ting-avg_rate_mean))
pre_rate_test_step2 <- avg_rate_mean+test %>% left_join(bias_movie_train,by = "movieI
d")%>%.$movie_bias_rate
step2_rsme <- RMSE(pre_rate_test_step2, test$rating)
step2_rsme
```

```
## [1] 0.9437429
```

The RSME for step 1 is 0.9437429 which is still too high.

Step 3 Add User Bias to the model

```
bias_user_train <- train%>%left_join(bias_movie_train,by = "movieId")%>%group_by(user
Id) %>% summarize(user_bias_rate = mean(rating-avg_rate_mean-movie_bias_rate))
pre_rate_test_step3 <- pre_rate_test_step2 + test %>% left_join(bias_movie_train,by =
"movieId")%>% left_join(bias_user_train,by = "userId")%>%.$user_bias_rate
step3_rsme <- RMSE(pre_rate_test_step3, test$rating)
step3_rsme
```

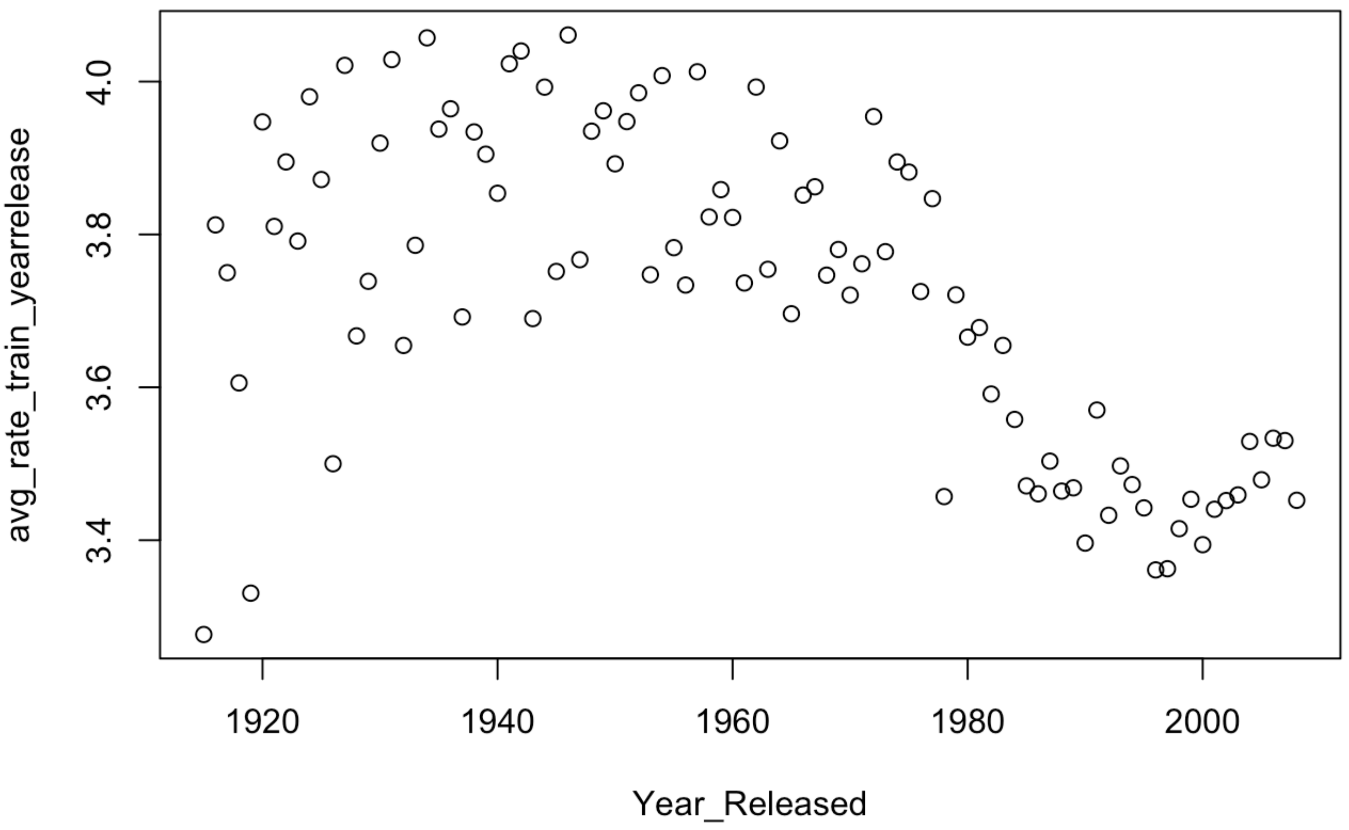
```
## [1] 0.865932
```

After adding the user bias to the model, the RSME is lower.

Step 4 Add Years Released Bias to the model

Explore average rate by year released.

```
avg_train_yearrelease <- train%>%group_by(Year_Released)%>%summarize(avg_rate_train_y
earrelease = mean(rating))
plot(avg_train_yearrelease)
```



From above chart, we can see that, average rating for older movies are generally higher than the movies made in recent years. There're obviously bias in the released year.

Add Years Released Bias to Step 3 Model

```
bias_year_train <-train%>%left_join(bias_movie_train,by = "movieId")%>% left_join(bia
s_user_train,by = "userId")%>%group_by(Year_Released) %>% summarize(year_bias_rate =
mean(rating-avg_rate_mean-movie_bias_rate-user_bias_rate))
pre_rate_test_step4 <- test%>%left_join(bias_movie_train,by = "movieId")%>% left_join
(bias_user_train,by = "userId")%>% left_join(bias_year_train,by = "Year_Released")%>%
mutate(pre = avg_rate_mean+movie_bias_rate+user_bias_rate+year_bias_rate )%>%.$pre
step4_rsme <- RMSE(pre_rate_test_step4, test$rating)
step4_rsme
```

```
## [1] 0.8656117
```

After adding the released year bias, though the RSME is a little lower, it is not significantly lower than Step3, only decreased roughly 0.03%. Therefore, try adding the movie genre bias instead of the year released bias, and see how's the model's performance.

Step 5 Add Movie Genres Bias instead of Year Released Bias to Step 3 Model

```
bias_genre_train <-train%>%left_join(bias_movie_train,by = "movieId")%>% left_join(bi
as_user_train,by = "userId")%>%group_by(genres) %>% summarize(genre_bias_rate = mean(
rating-avg_rate_mean-movie_bias_rate-user_bias_rate))
pre_rate_test_step5 <- test%>%left_join(bias_movie_train,by = "movieId")%>% left_join
(bias_user_train,by = "userId")%>% left_join(bias_genre_train,by = "genres")%>%mutate
(pre = avg_rate_mean+movie_bias_rate+user_bias_rate+genre_bias_rate )%>%.$pre
step5_rsme <- RMSE(pre_rate_test_step5, test$rating)
step5_rsme
```

```
## [1] 0.8655941
```

The RSME is slightly lower, but not significantly.

Step 6 Add all movies, users, year released and movie genres bias

```
pre_rate_test_step6 <- test%>%left_join(bias_movie_train,by = "movieId")%>% left_join
(bias_user_train,by = "userId")%>% left_join(bias_year_train,by = "Year_Released")%>%
left_join(bias_genre_train,by = "genres")%>%mutate(pre = avg_rate_mean+movie_bias_rat
e+user_bias_rate+year_bias_rate+genre_bias_rate )%>%.$pre
step6_rsme <- RMSE(pre_rate_test_step6, test$rating)
step6_rsme
```

```
## [1] 0.8654578
```

Print the RSME result for all steps

```
modelname<- c("Average","Movie Effect","Movie User Effect","Movie User Year Combo Eff
ect", "Movie User Genre Combo Effect","Movie User Year Genre Combo Effect" )
rsme_result <- data.frame(model=modelname,rsme= c(step1_rsme,step2_rsme,step3_rsme,st
ep4_rsme,step5_rsme,step6_rsme))
print(rsme_result)
```

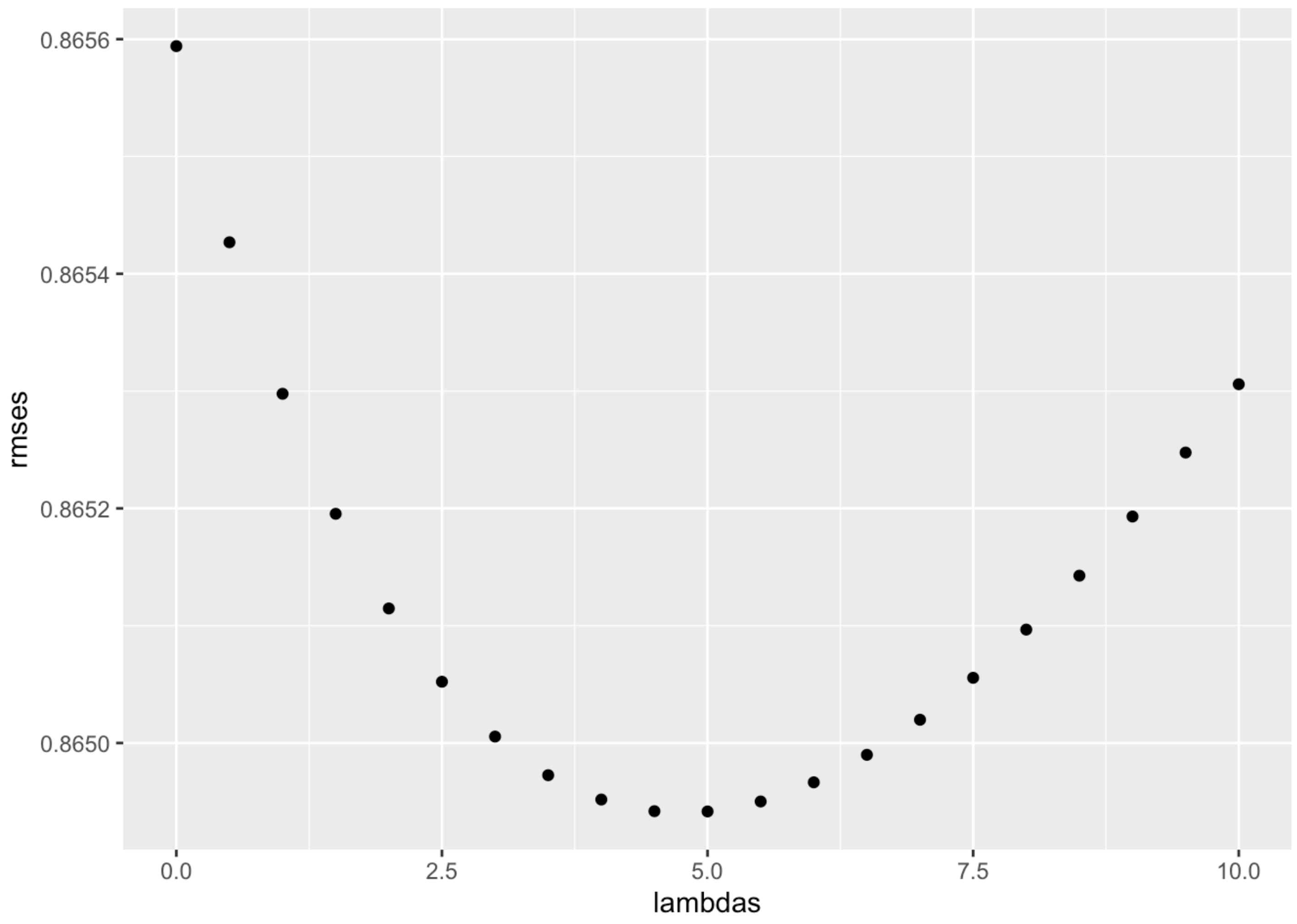
```
##                                model      rsme
## 1                            Average 1.0599043
## 2                       Movie Effect 0.9437429
## 3                  Movie User Effect 0.8659320
## 4       Movie User Year Combo Effect 0.8656117
## 5      Movie User Genre Combo Effect 0.8655941
## 6 Movie User Year Genre Combo Effect 0.8654578
```

From the table above, we can see that, from the six models, the performance of “Movie User Genre Combo Effect” is better. Though the rsme of “Movie User Year Genre Effect” model is slightly lower, it contains four components which is more complicated. Therefore, the “Movie User Year Genre Effect” model is selected for later analysis.

Regularization

Conduct regularization on model 5 “Movie User Genre Combo Effect”

```
lambdas <- seq(0,10,.5)
rmses <- sapply(lambdas, function(l){
  avg_rate_mean <- mean(train$rating)
  bias_movie_train <- train%>%group_by(movieId) %>% summarize(movie_bias_rate = sum(r
ating-avg_rate_mean)/(n()+l))
  bias_user_train <- train%>%left_join(bias_movie_train,by = "movieId")%>%group_by(us
erId) %>% summarize(user_bias_rate = sum(rating-avg_rate_mean-movie_bias_rate)/(n()+l
))
  bias_genre_train <-train%>%left_join(bias_movie_train,by = "movieId")%>% left_join(
bias_user_train,by = "userId")%>%group_by(genres) %>% summarize(genre_bias_rate = sum
(rating-avg_rate_mean-movie_bias_rate-user_bias_rate)/(n()+l))
  pre_rate <- test%>%left_join(bias_movie_train,by = "movieId")%>% left_join(bias_use
r_train,by = "userId")%>% left_join(bias_genre_train,by = "genres")%>%mutate(pre = av
g_rate_mean+movie_bias_rate+user_bias_rate+genre_bias_rate)%>%.$pre
  return(RMSE(pre_rate, test$rating))
})
qplot(lambdas,rmses)
```



```
lambdas[which.min(rmses)]
```

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

When lambda is 5, the RMSE for the chosen model is the smallest.

Part 5 Model Validation

Now validate the model with the validation dataset

```
l <- 5
avg_rate_va <- mean(validation$rating)
bias_movie_va <- validation%>%group_by(movieId) %>% summarize(movie_bias_rate_va = su
m(rating-avg_rate_va)/(n()+l))
bias_user_va <- validation%>%left_join(bias_movie_va,by = "movieId")%>%group_by(userI
d) %>% summarize(user_bias_rate_va = sum(rating-avg_rate_va-movie_bias_rate_va)/(n()+
l))
bias_genre_va <-validation%>%left_join(bias_movie_va,by = "movieId")%>% left_join(bia
s_user_va,by = "userId")%>%group_by(genres) %>% summarize(genre_bias_rate_va = sum(ra
ting-avg_rate_va-movie_bias_rate_va-user_bias_rate_va)/(n()+l))
pre_rate_va <- validation%>%left_join(bias_movie_va,by = "movieId")%>% left_join(bias
_user_va,by = "userId")%>% left_join(bias_genre_va,by = "genres")%>%mutate(pre = avg_
rate_mean+movie_bias_rate_va+user_bias_rate_va+genre_bias_rate_va)%>%.$pre
RMSE(pre_rate_va, validation$rating)
```

```
## [1] 0.8399034
```

After applying the model to the validation dataset, we can get the RMSE is 0.8399034.

Part 6 Conclusion

To conclude, since the dataset contains 10M data, it will be difficult to directly use linear model lm() to fit a model, therefore, I tidied the dataset, and conducted exploratory analysis first, and tried to get more information about the dataset first. Then I added the variable bias to the average linear model one by one, and calculate the RMSE respectively, so as to find out how the components can explain the dataset. For the better performance, I also conducted regularization on the chosen model to get the better result. Finally, I used the validation dataset to validate the model.

This project helps me to enhance my skills in exploratory analysis using R, and trained me how to get a better linear model for a large dataset. I didn’t use matrix factorization for this project, I will try to apply matrix factorization method later in the future, and compare the results.

Last but not the least, thank HarvardX for providing such nice courses online and benefit us all!