MovieLensProject

Bita Parsa

14/11/2021

Capstone MovieLens Project

Introduction

Many companies like Amazon, Netflix, use ratings or stars that users will give to a specific item. They can collect a large amount of data to predict how many stars or ratings a user will give to a specific item. Here we want to use GroupLens research lab database to generate a recommendation system for Netflix.This project aims to create a machine learning model to predict how a user rates a movie. In this project, we will try to increase the accuracy of Netflix recommendation system by using some strategies.

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

We will use the 10M version of the MovieLens dataset downloaded from here:

"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")

# if using R 4.0 or later:
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(movieId),
                                           title = as.character(title),
                                           genres = as.character(genres))


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

The algorithm develops using the edx set. We will use validation set only for the final test since they are assumed are unknown.

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

edx <- movielens[-test_index,]
temp <- movielens[test_index,]

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

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

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

Data Exploration

```r
#Project

#Exploring Data

head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     292      5 838983421               Outbreak (1995)
## 4:      1     316      5 838983392               Stargate (1994)
## 5:      1     329      5 838983392 Star Trek: Generations (1994)
## 6:      1     355      5 838984474       Flintstones, The (1994)
##                           genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:  Action|Drama|Sci-Fi|Thriller
## 4:       Action|Adventure|Sci-Fi
## 5: Action|Adventure|Drama|Sci-Fi
## 6:       Children|Comedy|Fantasy
```

```r
glimpse(edx)
```

```
## Rows: 9,000,055
## Columns: 6
## $ userId    <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, ~
## $ movieId   <dbl> 122, 185, 292, 316, 329, 355, 356, 362, 364, 370, 377, 420, ~
## $ rating    <dbl> 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, ~
## $ timestamp <int> 838985046, 838983525, 838983421, 838983392, 838983392, 83898~
```

```
## $ title     <chr> "Boomerang (1992)", "Net, The (1995)", "Outbreak (1995)", "S~
## $ genres    <chr> "Comedy|Romance", "Action|Crime|Thriller", "Action|Drama|Sci~
```

It shows we have 9,000,055 rows and 6 column (userId,movieId,rating,timestamp ,title ,genres).

```
#number of unique users provided  ratings and number of unique movies were rated

edx %>%
  summarize(n_user = n_distinct(userId), n_movie = n_distinct(movieId))
```

```
##   n_user n_movie
## 1  69878   10677
```

It shows the number of unique users and number of unique movies.

```
# Separating the year of released from the title

pattern <- "[/(]\\d{4}[/)]$"
st_year_p<- str_extract(edx$title, pattern)
st_year <- str_extract(st_year_p, regex("\\d{4}"))


t <- str_remove(edx$title, pattern)

edxnew <- edx %>% mutate(year_released = as.numeric(st_year), title = t)
```

We can separate the year of released from title.

```
#10 best movies (most rated with the highest rate)
top_movies <- edxnew %>%
  group_by(movieId) %>%
  summarize(n=n(), s=sum(rating), title = first(title)) %>%
  top_n(10, n*s)
top_movies
```

```
## # A tibble: 10 x 4
##    movieId     n       s title
##      <dbl> <int>   <dbl> <chr>
##  1     110 26212 106994. "Braveheart "
##  2     260 25672 108370. "Star Wars: Episode IV - A New Hope (a.k.a. Star Wars)~
##  3     296 31362 130302. "Pulp Fiction "
##  4     318 28015 124810. "Shawshank Redemption, The "
##  5     356 31079 124714. "Forrest Gump "
##  6     457 25998 104230  "Fugitive, The "
##  7     480 29360 107561  "Jurassic Park "
##  8     527 23193 101202. "Schindler's List "
##  9     589 25984 102062. "Terminator 2: Judgment Day "
## 10     593 30382 127729  "Silence of the Lambs, The "
```

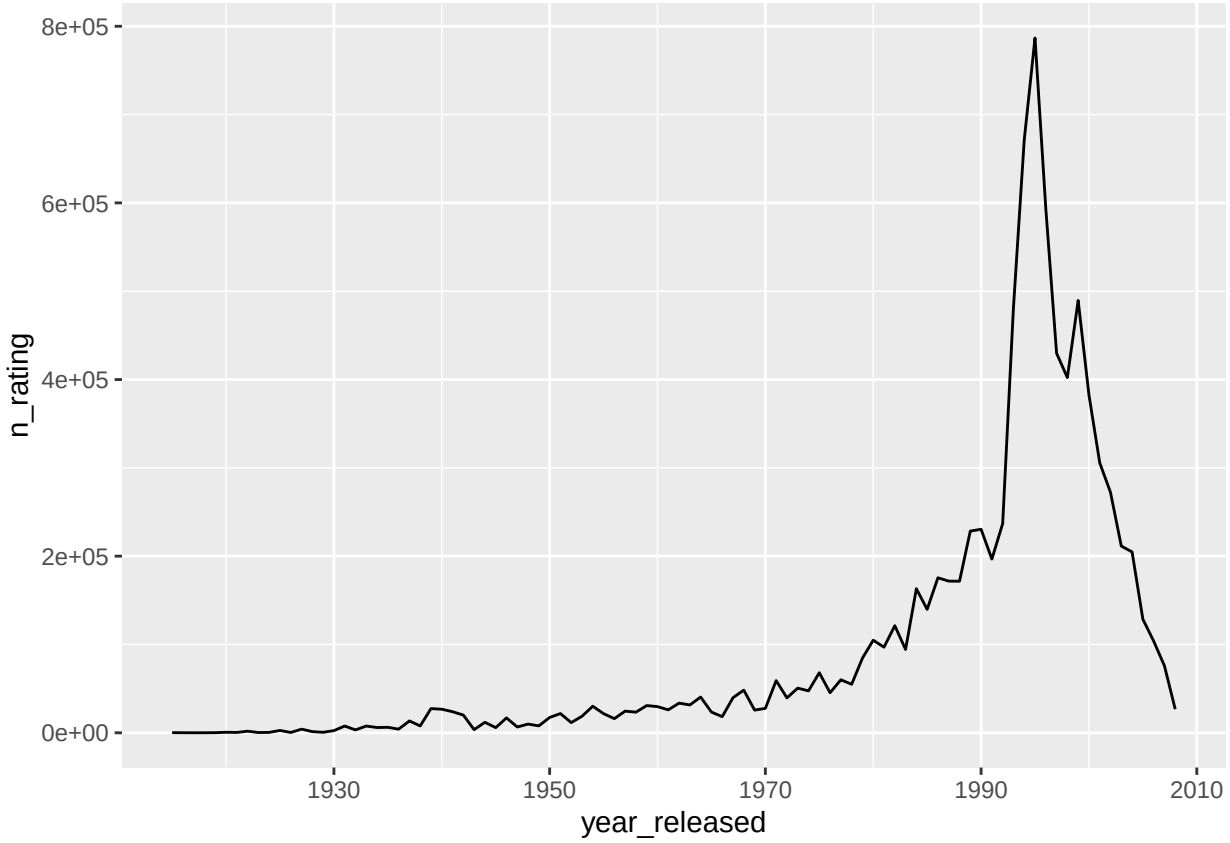
These are the 10 best movies

```
# plot number of rating for each year of released
movie <- edxnew %>%
  select(movieId,year_released) %>%
  group_by(year_released) %>%
  summarise(n_rating=n())

movie$year_released[which.max(movie$n_rating)]
```

```
## [1] 1995
```

We have the max number of rating for the movies with the release year of 1995.

```
movie %>%
  ggplot(aes(year_released,n_rating )) +
  geom_line()
```



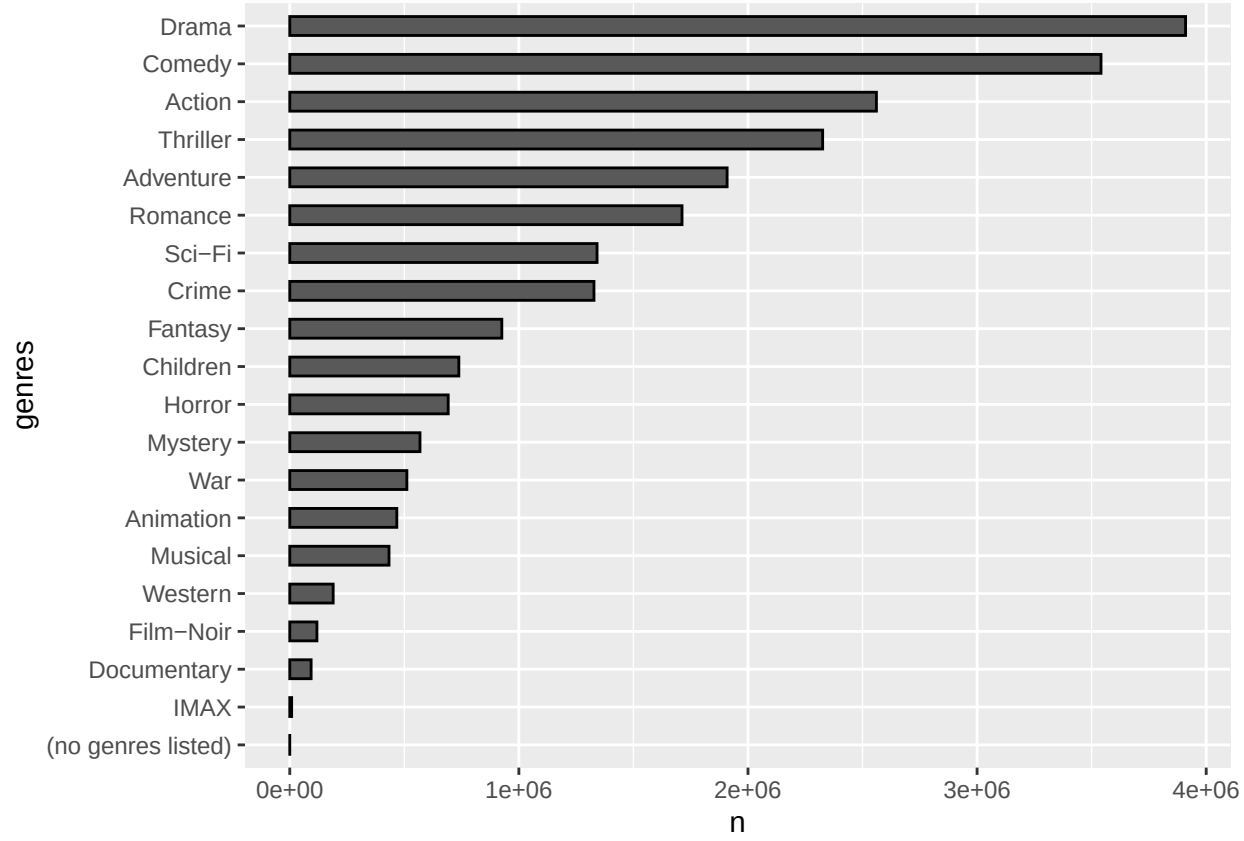
Plot shows number of rating for each year of released.

```
#Number of movies for each genres
movie_generes <- edxnew %>% separate_rows(genres,sep="\\|")
genres <- levels(as.factor(movie_generes$genres))

sapply(genres, function(g) {
  sum(str_detect(edx$genres, g))
})
```

```
## (no genres listed)            Action         Adventure         Animation
##                  7           2560545           1908892            467168
##           Children            Comedy             Crime       Documentary
##             737994           3540930           1327715             93066
##              Drama           Fantasy         Film-Noir            Horror
##            3910127            925637            118541            691485
##               IMAX           Musical           Mystery           Romance
##               8181            433080            568332           1712100
##             Sci-Fi          Thriller               War           Western
##            1341183           2325899            511147            189394
```

It shows number of movies for each genres.

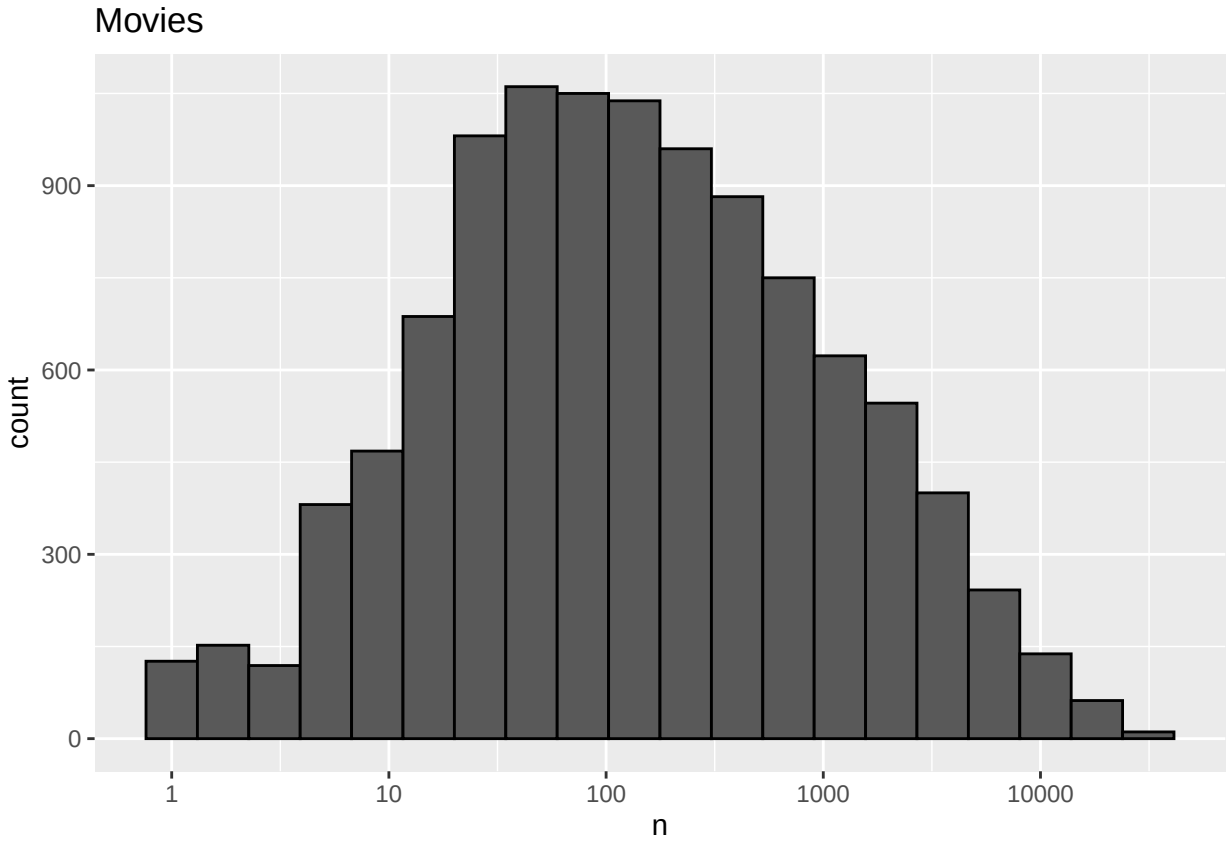
```
# which genres is rated more
movie_generes%>%
  group_by(genres)%>%
  summarise(n=n())%>%
  mutate(genres=reorder(genres, n))%>%
  ggplot(aes(genres, n)) +
  geom_bar(width = 0.5, stat = "identity", color = "black") +
  coord_flip()
```



The bar plot shows that Drama genre movies are rated more.

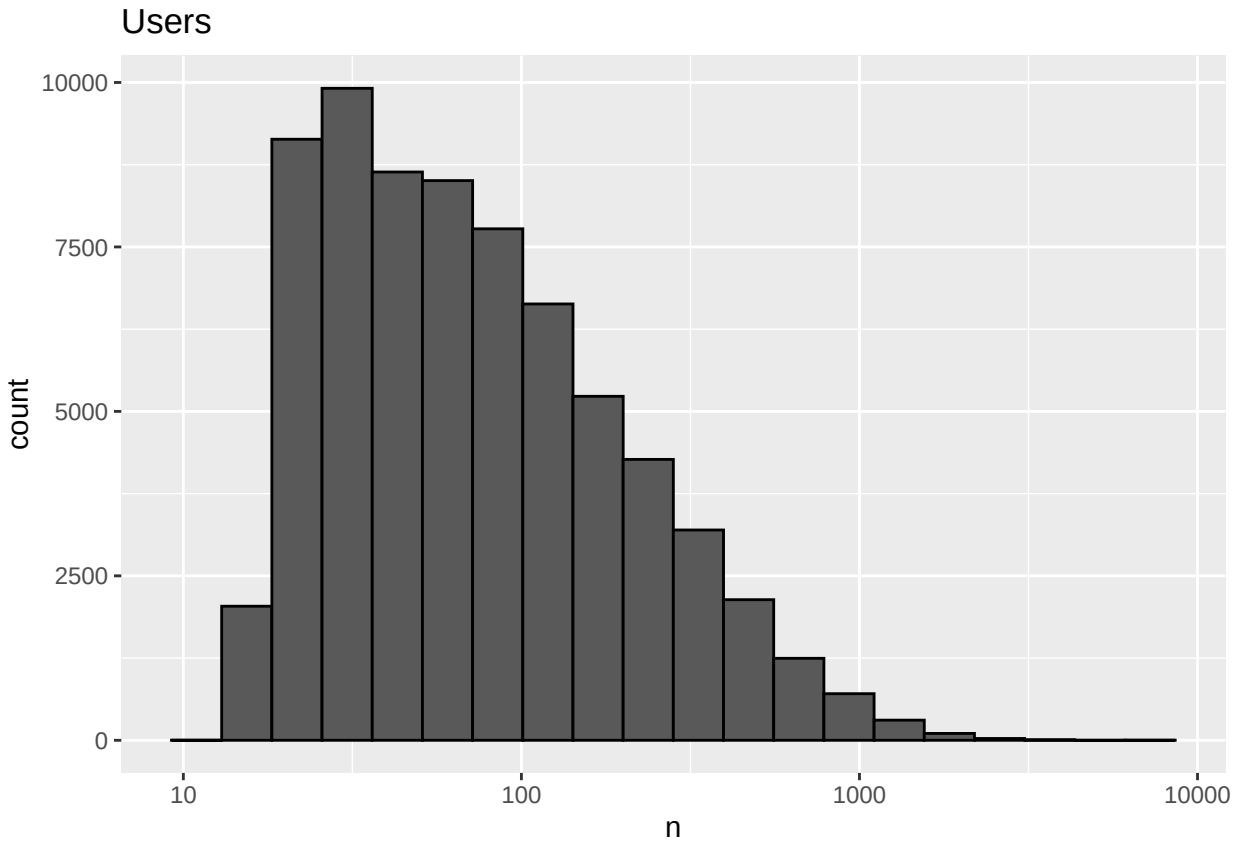
Now we want to look at some of the general properties of the data.

```r
# Distribution of movies getting rated

edxnew %>% group_by(movieId)%>% summarise(n=n())%>%
  ggplot(aes(n)) +
  geom_histogram(bins = 20, color = "black") +
  scale_x_log10() +
  ggtitle("Movies")
```



The above bar plot shows the distribution of movies getting rated and how some of the movies are getting rated more than others.

```r
# Distribution of users rating the movies

edxnew %>% group_by(userId)%>% summarise(n=n())%>%
  ggplot(aes(n)) +
  geom_histogram(bins = 20, color = "black") +
  scale_x_log10() +
  ggtitle("Users")
```



The distribution of users rating the movies shows that some of the users rated more than others.

Recommendation system for prediction

To create a recommendation system as a machine learning challenge, at first we need to generate the test set and the train set.

```
#Create test set and train set

set.seed(1)
test_index <- createDataPartition(y=edxnew$rating, times=1, p=0.1, list=FALSE)

train_set <- edxnew[-test_index,]
test_set_temp <- edxnew[test_index,]
```

To ensure that we do not have users and movies in the test set that do not exist in the train set, we need to remove these entries by using the semi_join function.

```
# Make sure userId and movieId in test_set  are also in train_set_temp set
test_set <- test_set_temp %>%
  semi_join(train_set, by = "movieId") %>%
  semi_join(train_set, by = "userId")
```

And then, we add all those removed entries to the train set.

```
# Add rows removed from test_set  back into train_set
removed <- anti_join(test_set_temp, test_set)
train_set <- rbind(train_set, removed)
```

We can remove extra object from memory.

```
# Delete extra object from memory
rm(test_index,test_set_temp, removed)
```

We want to find an accurate recommendation system, so the less RMSE(residual mean squared error) gives us, the more accurate system.The RMSE is defined as below.

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

N = The number of observations

$\hat{y}_{u,i}$= predicted user rating

$y_{u,i}$= actual rating for movie i by user u

```
#function that computes the RMSE for vectors of ratings and corresponding predictors.

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

Model No.1- Naive Model

The first model is very simple, and we can predict the rating of all the movies are the same and equal to the average of the rating of the movies.

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

- μ the average of all movie ratings
- $\epsilon_{u,i}$ independently errors sample from the same distribution centered at 0 and μ

```
#Model No.1 (Average of Rating-Naive Model)

mu <- mean(train_set$rating)
naive_rmse <- RMSE(test_set$rating,mu)
```

Now we can create a result table and start with the naive result.

```
#Creating a result table
RMSE_results <- tibble(method ="Average of Rating", RMSE = naive_rmse)
```

Model No.2- Movie Effects Model

We found out that some of the movies are rated higher that the other and different movies are rated differently,so we can improve our system by adding b_{i}:

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

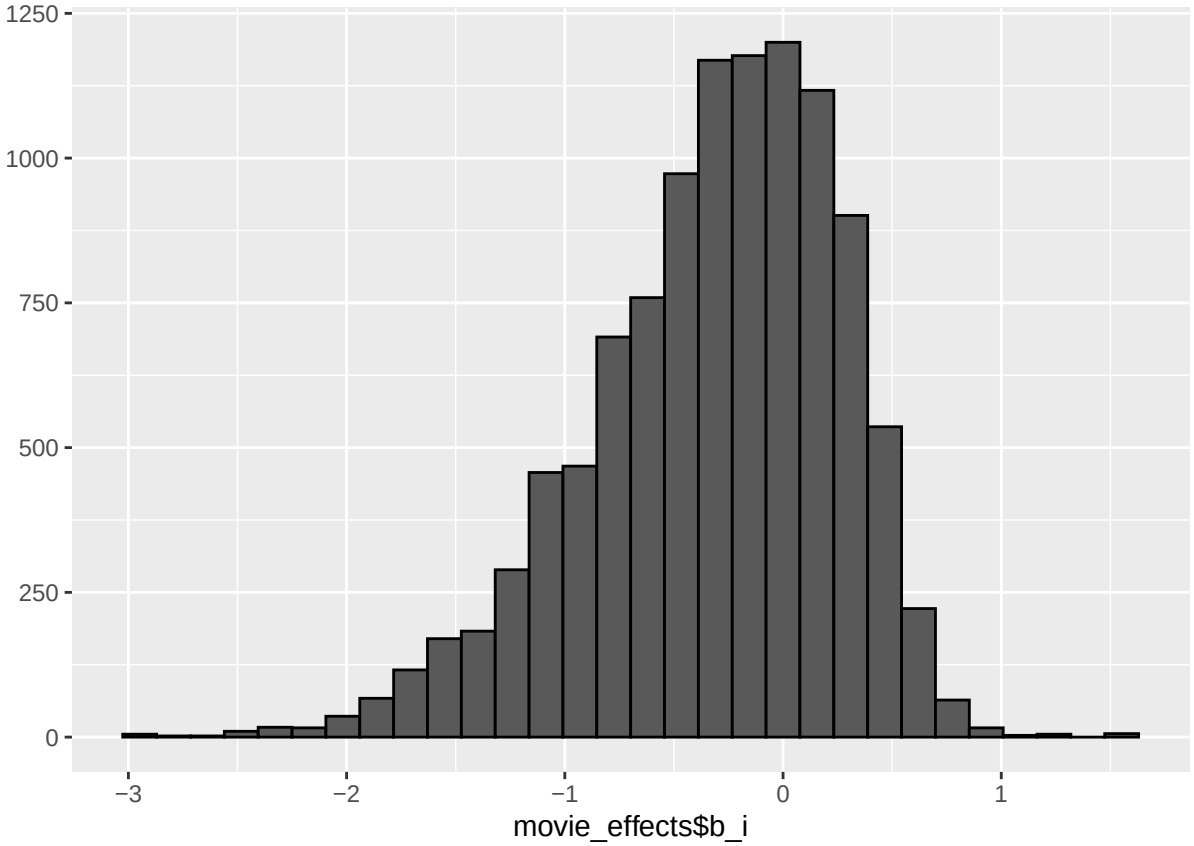
- b_i average ranking for movie i

We know that the least squares estimate $\hat{b}_i$ is the average of $Y_{u,i}$-$\hat{\mu}$ So we can calculate $\hat{b}_i$ from below:

```
#Model No.2 (Movie Effect b_i)




movie_effects <- train_set %>%
  group_by(movieId)%>%
summarize(b_i= mean(rating-mu))
```

below we can see our estimation for $\hat{b}_i$

```
# We can see our estimates
qplot(movie_effects$b_i,  bins = 30, color = I("black"))
```



Now we can calculate the predicted rating and then the RMSE.

```
b_i_test <- test_set %>%
  left_join(movie_effects, by="movieId") %>%
  pull(b_i)
```

```
predicted_ratings <- mu + b_i_test


RMSE_model <- RMSE(test_set$rating, predicted_ratings)

RMSE_results <- bind_rows(RMSE_results,
                          data_frame(method="Movie Effect Model",RMSE = RMSE_model ))

RMSE_results %>% knitr::kable()
```

method	RMSE
Average of Rating	1.0600537
Movie Effect Model	0.9429615

The result shows that the accuracy improved.

Model No.3- User Effects Model

It is obvious that some of the users are fastidious and some of them easygoing, so we can add effects of user to our equation to improve the result.

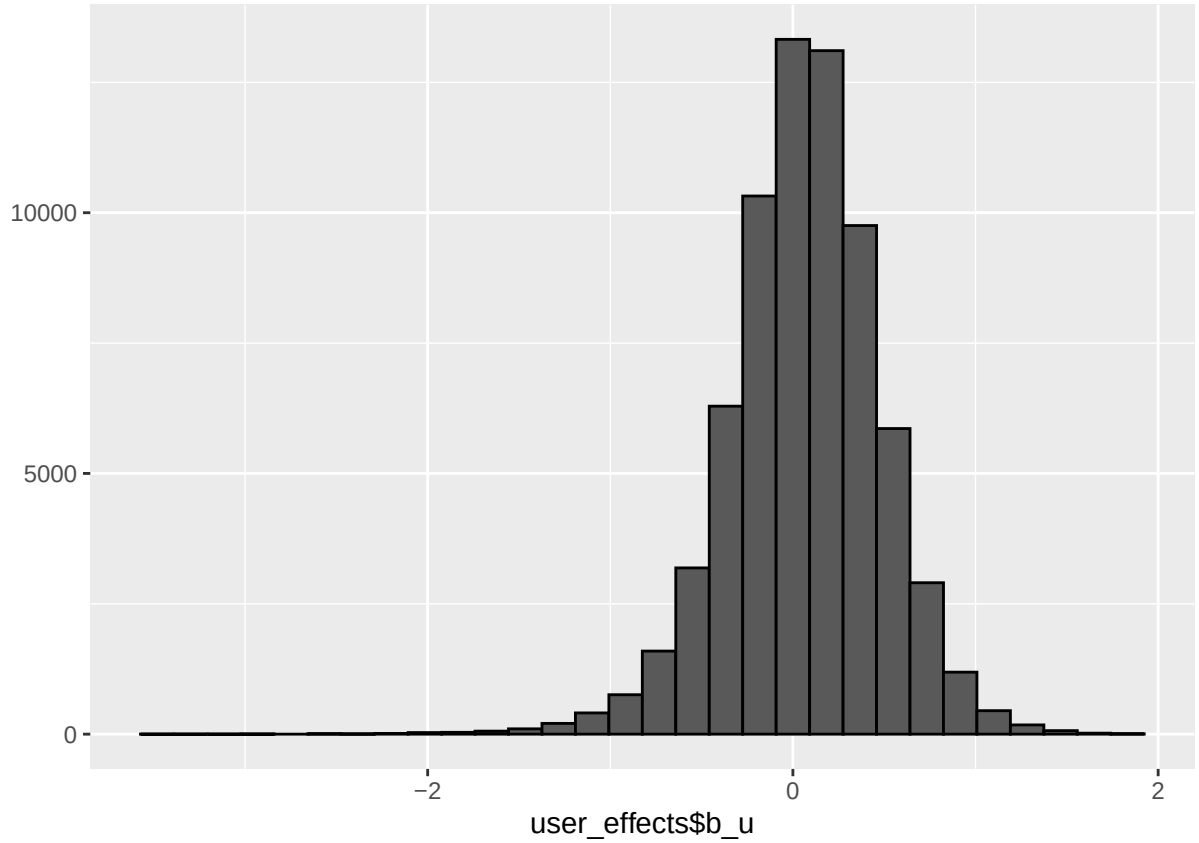
$$Y_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

- b_u average rating for user u

```
#Model No.3 (User Effect b_u)

user_effects <- train_set %>%
  left_join(movie_effects, by="movieId")%>%
  group_by(userId)%>%
  summarize(b_u= mean(rating-mu-b_i))
```

Below we can see our estimation for $\hat{b}_u$

```
# We can see our estimates
qplot(user_effects$b_u,  bins = 30, color = I("black"))
```



Now we can calculate the predicted rating and then the RMSE.

```
b_u_test <- test_set %>%
  left_join(user_effects,by="userId")%>%
  pull(b_u)

predicted_ratings <- mu + b_i_test + b_u_test

RMSE_model <- RMSE(test_set$rating, predicted_ratings)

RMSE_results <- bind_rows(RMSE_results,
                          data_frame(method="Movie Effect and User Effect Model",RMSE = RMSE_model ))

RMSE_results %>% knitr::kable()
```

method	RMSE
Average of Rating	1.0600537
Movie Effect Model	0.9429615
Movie Effect and User Effect Model	0.8646843

So we can see that we have significant improvement in the result.

Model No.4- Genres Effects Model

Some of the genres are more popular so rated more than the others, we can add effects of genres to our equation to find more accurate system.

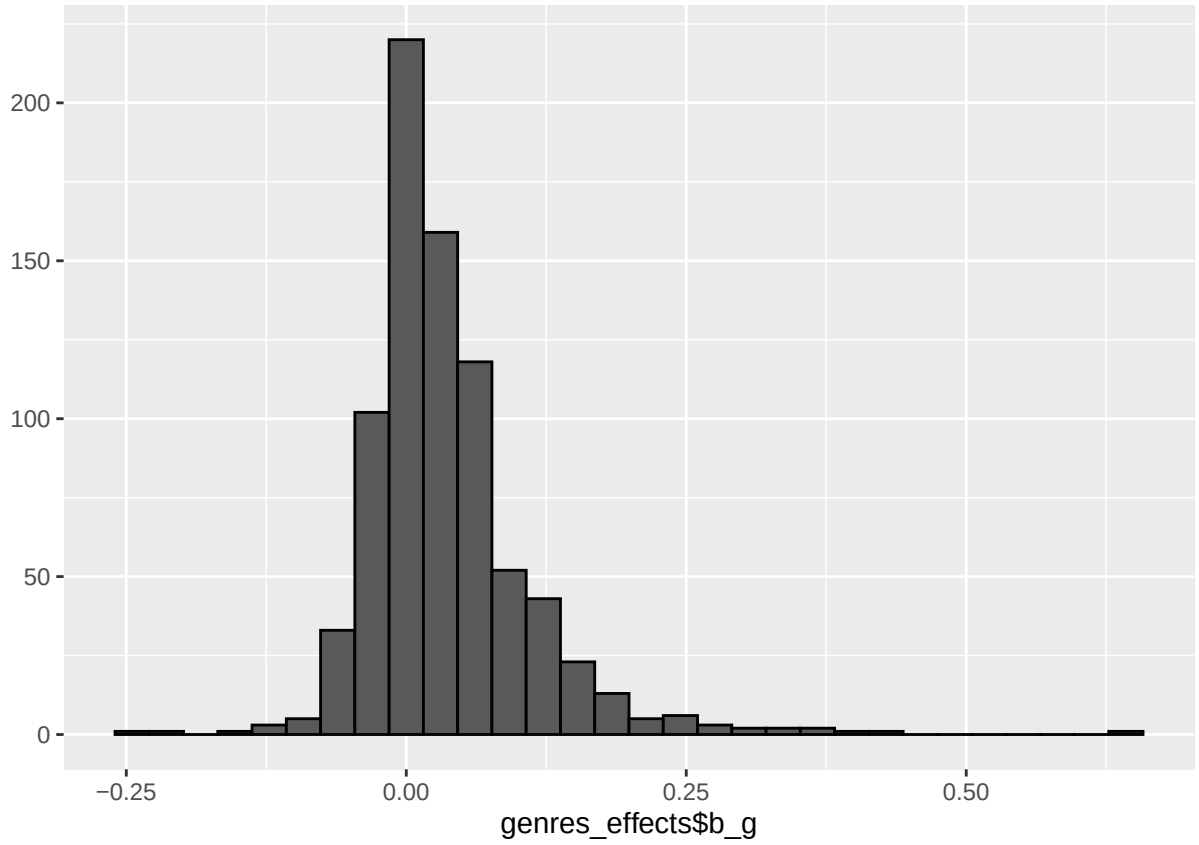
$$Y_{u,i} = \mu + b_i + b_u + b_g + \epsilon_{u,i}$$

- b_g average rating for each genres

```
#Model No.4 (Genres Effect b_g)

genres_effects <- train_set %>%
  left_join(movie_effects, by="movieId")%>%
  left_join(user_effects, by="userId")%>%
  group_by(genres)%>%
  summarize(b_g= mean(rating-mu-b_i-b_u))
```

Below we can see our estimation for $\hat{b}_g$

```
qplot(genres_effects$b_g,  bins = 30, color = I("black"))
```



Now we can calculate the predicted rating and then the RMSE.

```
b_g_test <- test_set %>%
  left_join(genres_effects,by="genres")%>%
  pull(b_g)
```

```
predicted_ratings <- mu + b_i_test + b_u_test + b_g_test

RMSE_model <- RMSE(test_set$rating, predicted_ratings)

RMSE_results <- bind_rows(RMSE_results,
                          data_frame(method="Movie Effect and User Effect and Genres Effect Model",RMSE

RMSE_results %>% knitr::kable()
```

method	RMSE
Average of Rating	1.0600537
Movie Effect Model	0.9429615
Movie Effect and User Effect Model	0.8646843
Movie Effect and User Effect and Genres Effect Model	0.8643241

It seems that our system improved but not as much as before.

Regularization

Now we want to explore where we made mistakes in the Model No.2 (Movie Effects Model)
Below are the list of the largest mistake.

```
# We made a mistake in the model No.2
test_set %>%
  left_join(movie_effects, by="movieId") %>%
  mutate(residual=rating-(b_i+mu)) %>%
  arrange(desc(abs(residual))) %>%
  select(title,residual,b_i)%>%
  slice (1:10)
```

```
##                          title  residual        b_i
##  1:      From Justin to Kelly   4.125683 -2.6381387
##  2: Shawshank Redemption, The  -3.956567  0.9441110
##  3: Shawshank Redemption, The  -3.956567  0.9441110
##  4:            Godfather, The  -3.916651  0.9041954
##  5:            Godfather, The  -3.916651  0.9041954
##  6:            Godfather, The  -3.916651  0.9041954
##  7:            Godfather, The  -3.916651  0.9041954
##  8:       Usual Suspects, The  -3.866552  0.8540962
##  9:          Schindler's List  -3.864085  0.8516292
## 10:          Schindler's List  -3.864085  0.8516292
```

Below are the list of 10 best and 10 worst movies according to our estimate:

```
#10 best movie according to our estimate
train_set %>%
  select(movieId,title) %>%
  distinct() %>%
  left_join(movie_effects, by="movieId") %>%
```

```
    arrange(desc(b_i)) %>%
  select(title, b_i) %>%
  slice(1:10)
```

```
##                                                                                 title
##  1:                                                       Sun Alley (Sonnenallee)
##  2:                                                 Fighting Elegy (Kenka erejii)
##  3:                                                 Satan’s Tango (SÃ¡tÃ¡ntangÃ³)
##  4:                                               Shadows of Forgotten Ancestors
##  5:                                                       Hellhounds on My Trail
##  6:                                          Blue Light, The (Das Blaue Licht)
##  7: Who’s Singin’ Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva)
##  8:                                 Life of Oharu, The (Saikaku ichidai onna)
##  9:                                                       Constantine’s Sword
## 10:                              Human Condition II, The (Ningen no joken II)
##          b_i
##  1: 1.487544
##  2: 1.487544
##  3: 1.487544
##  4: 1.487544
##  5: 1.487544
##  6: 1.487544
##  7: 1.237544
##  8: 1.237544
##  9: 1.237544
## 10: 1.237544
```

```
#10 worst movie according to our estimate
train_set %>%
  select(movieId,title) %>%
  distinct() %>%
  left_join(movie_effects, by="movieId") %>%
  arrange(b_i) %>%
  select(title , b_i) %>%
  slice(1:10)
```

```
##                                 title       b_i
##  1:                          Besotted  -3.012456
##  2: War of the Worlds 2: The Next Wave  -3.012456
##  3:         Confessions of a Superhero  -3.012456
##  4:                       Hi-Line, The  -3.012456
##  5:                 Accused (Anklaget)  -3.012456
##  6:       SuperBabies: Baby Geniuses 2  -2.767775
##  7:                     Disaster Movie  -2.745789
##  8:               From Justin to Kelly  -2.638139
##  9:                  Hip Hop Witch, Da  -2.603365
## 10:                      Roller Boogie  -2.512456
```

It seems that they are obscure movies, so it is better to find how much these two group rated.

```
#we want to find how much these two group rated
# we can find the times of rating for each movieId
```

```
train_set_n_rating <- train_set %>%
  count(movieId) %>%
  select(movieId,n)

# number of rating for 10 best movies according to our estimate
train_set %>%
  select(movieId,title) %>%
  distinct() %>%
  left_join(movie_effects, by="movieId") %>%
  arrange(desc(b_i)) %>%
  select(title, movieId) %>%
  slice(1:10) %>%
  left_join(train_set_n_rating)
```

```
##                                                                                 title
##  1:                                                      Sun Alley (Sonnenallee)
##  2:                                                  Fighting Elegy (Kenka erejii)
##  3:                                                  Satan’s Tango (SÃ¡tÃ¡ntangÃ³)
##  4:                                                 Shadows of Forgotten Ancestors
##  5:                                                         Hellhounds on My Trail
##  6:                                               Blue Light, The (Das Blaue Licht)
##  7: Who’s Singin’ Over There? (a.k.a. Who Sings Over There) (Ko to tamo peva)
##  8:                                    Life of Oharu, The (Saikaku ichidai onna)
##  9:                                                        Constantine’s Sword
## 10:                                 Human Condition II, The (Ningen no joken II)
##     movieId n
##  1:   53355 1
##  2:   51209 1
##  3:   33264 1
##  4:   42783 1
##  5:    3226 1
##  6:   64275 1
##  7:    5194 4
##  8:   25975 2
##  9:   65001 2
## 10:   26048 4
```

```
# number of rating for 10 worst movies according to our estimate
train_set %>%
  select(movieId,title) %>%
  distinct() %>%
  left_join(movie_effects, by="movieId") %>%
  arrange(b_i) %>%
  select(title ,movieId) %>%
  slice(1:10)%>%
  left_join(train_set_n_rating)
```

```
##                                  title movieId   n
##  1:                          Besotted     5805   1
##  2: War of the Worlds 2: The Next Wave    64999   2
##  3:         Confessions of a Superhero    63828   1
##  4:                        Hi-Line, The     8394   1
##  5:                Accused (Anklaget)    61768   1
```

```
##  6:       SuperBabies: Baby Geniuses 2     8859  47
##  7:                     Disaster Movie    61348  30
##  8:              From Justin to Kelly     6483 183
##  9:                 Hip Hop Witch, Da     7282  11
## 10:                     Roller Boogie     8856  13
```

We can see that most of them rated few times and they are quite obscure.

We need to regularize to constrain the total variability of the effect sizes. Assume we have movie i=1 with 100 ratings and movie i=2,3,4,5 with only 1 rating. If we use least squares, for the first movie effect, b_1 is the average of the 100 ratings, and for movies 2,3,4,5, the average rating is the estimation based on just one number, so it is inaccurate.
The general idea of regularization is to control the total variability of the movie effect, so we try to minimize an equation that adds a penalty.

$$1/N \sum_{u,i} (y_{u,i} - \mu - b_i)^2 - \lambda \sum_i b_i^2$$

First term is the least squares and the second is a penalty that it depends on the b_i

$$\hat{b}_i(\lambda) = 1/(\lambda + n_i) \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu})$$

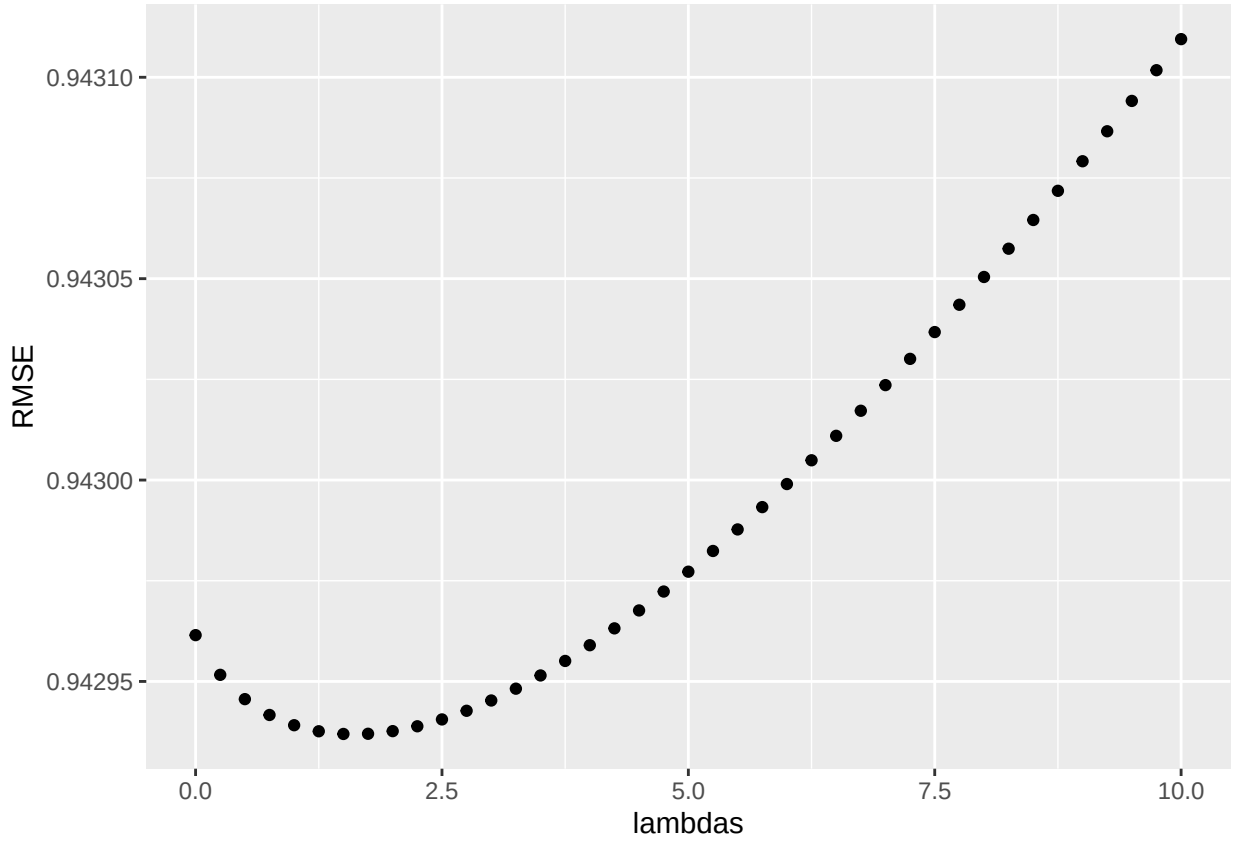
n_i number of ratings made for movie i when the sample size is large the penalty λ is ignored since $n_i + \lambda$ is very close to n_i, and when the n_i is small then $\hat{b}_i(\lambda)$ is shrunken to 0.

Model No.5 Regularized Movie Effect Model

We can use the cross validation to find the best amount for λ which give us the minimum amount of RMSE.

```
# Model No.5  Regularized Movie Effect Model

mu <- mean(train_set$rating)

lambdas <- seq(0, 10, 0.25)


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

  movie_effects_reg <- train_set %>%
    group_by(movieId)%>%
    summarise(b_i= sum(rating-mu)/(n()+l),n_i=n())

    b_i_reg_test <- test_set %>%
    left_join(movie_effects_reg, by="movieId") %>%
    pull(b_i)

  predicted_ratings <- mu + b_i_reg_test


  RMSE_reg_test <- RMSE(test_set$rating, predicted_ratings)

return(RMSE_reg_test)
```

```
})



qplot(lambdas, RMSE)
```




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

```
## [1] 1.5
```

Now we can use the calculated λ from above to find the RMSE for regularized movie effect model.

```
lambdas <- lambdas[which.min(RMSE)]

movie_effects_reg <- train_set %>%
  group_by(movieId)%>%
  summarise(b_i= sum(rating-mu)/(n()+lambdas),n_i=n())

b_i_reg_test <- test_set %>%
  left_join(movie_effects_reg, by="movieId") %>%
  pull(b_i)

predicted_ratings <- mu + b_i_reg_test
```

```
RMSE_reg_test <- RMSE(test_set$rating, predicted_ratings)


RMSE_results <- bind_rows(RMSE_results,
                          data_frame(method="Regularized Movie Effect Model",RMSE = RMSE_reg_test ))

RMSE_results %>% knitr::kable()
```

method	RMSE
Average of Rating	1.0600537
Movie Effect Model	0.9429615
Movie Effect and User Effect Model	0.8646843
Movie Effect and User Effect and Genres Effect Model	0.8643241
Regularized Movie Effect Model	0.9429370

Model No.6 Regularized Movie Effect and User Effect Model

We can use regularization for user effects model too.

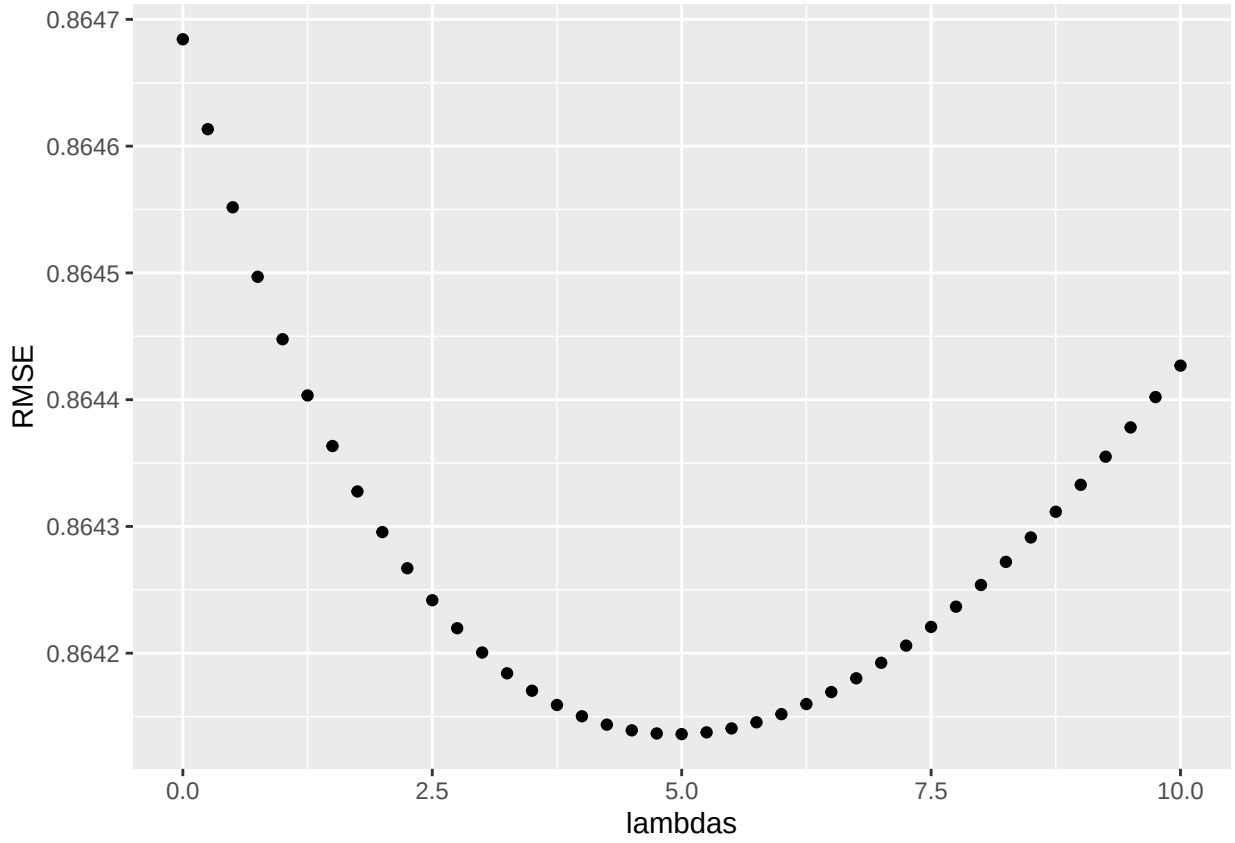
$$1/N \sum_{u,i}(y_{u,i} - \mu - b_i)^2 - \lambda(\sum_i b_i^2 + \sum_u b_u^2)$$

```
# Model No.6 Regularized Movie Effect and User Effect Model

mu <- mean(train_set$rating)

lambdas <- seq(0, 10, 0.25)

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

   movie_effects_reg <- train_set %>%
    group_by(movieId)%>%
    summarize(b_i= sum(rating-mu)/(n()+l),n_i=n())

  b_i_reg_test <- test_set %>%
    left_join(movie_effects_reg, by="movieId") %>%
    pull(b_i)

  user_effects_reg <- train_set %>%
    left_join(movie_effects_reg, by="movieId")%>%
    group_by(userId)%>%
    summarize(b_u= sum(rating-mu-b_i)/(n()+l) ,n_i=n())

  b_u_reg_test <- test_set %>%
    left_join(user_effects_reg, by="userId") %>%
    pull(b_u)


  predicted_ratings <- mu + b_i_reg_test + b_u_reg_test
```

```
  RMSE_reg_test <- RMSE(test_set$rating, predicted_ratings)
  return(RMSE_reg_test)

  })
```

Now we will use the calculated λ from above to find the RMSE for regularized movie effect and user effect model.

```
qplot(lambdas, RMSE)
```




```
lambdas <- lambdas[which.min(RMSE)]


mu <- mean(train_set$rating)
movie_effects_reg <- train_set %>%
  group_by(movieId)%>%
  summarize(b_i= sum(rating-mu)/(n()+lambdas),n_i=n())

b_i_reg_test <- test_set %>%
  left_join(movie_effects_reg, by="movieId") %>%
  pull(b_i)

user_effects_reg <- train_set %>%
  left_join(movie_effects_reg, by="movieId")%>%
  group_by(userId)%>%
```

```
  summarize(b_u= sum(rating-mu-b_i)/(n()+lambdas) ,n_i=n())

b_u_reg_test <- test_set %>%
  left_join(user_effects_reg, by="userId") %>%
  pull(b_u)


predicted_ratings <- mu + b_i_reg_test + b_u_reg_test


RMSE_reg_test <- RMSE(test_set$rating, predicted_ratings)


RMSE_results <- bind_rows(RMSE_results,
                          data_frame(method="Regularized Movie Effect and User Effect Model",RMSE = RMSE

RMSE_results %>% knitr::kable()
```

method	RMSE
Average of Rating	1.0600537
Movie Effect Model	0.9429615
Movie Effect and User Effect Model	0.8646843
Movie Effect and User Effect and Genres Effect Model	0.8643241
Regularized Movie Effect Model	0.9429370
Regularized Movie Effect and User Effect Model	0.8641362

Model No.7 Regularized Movie Effect and User Effect and Genres Effect Model

We can use regularization for user effects model too.

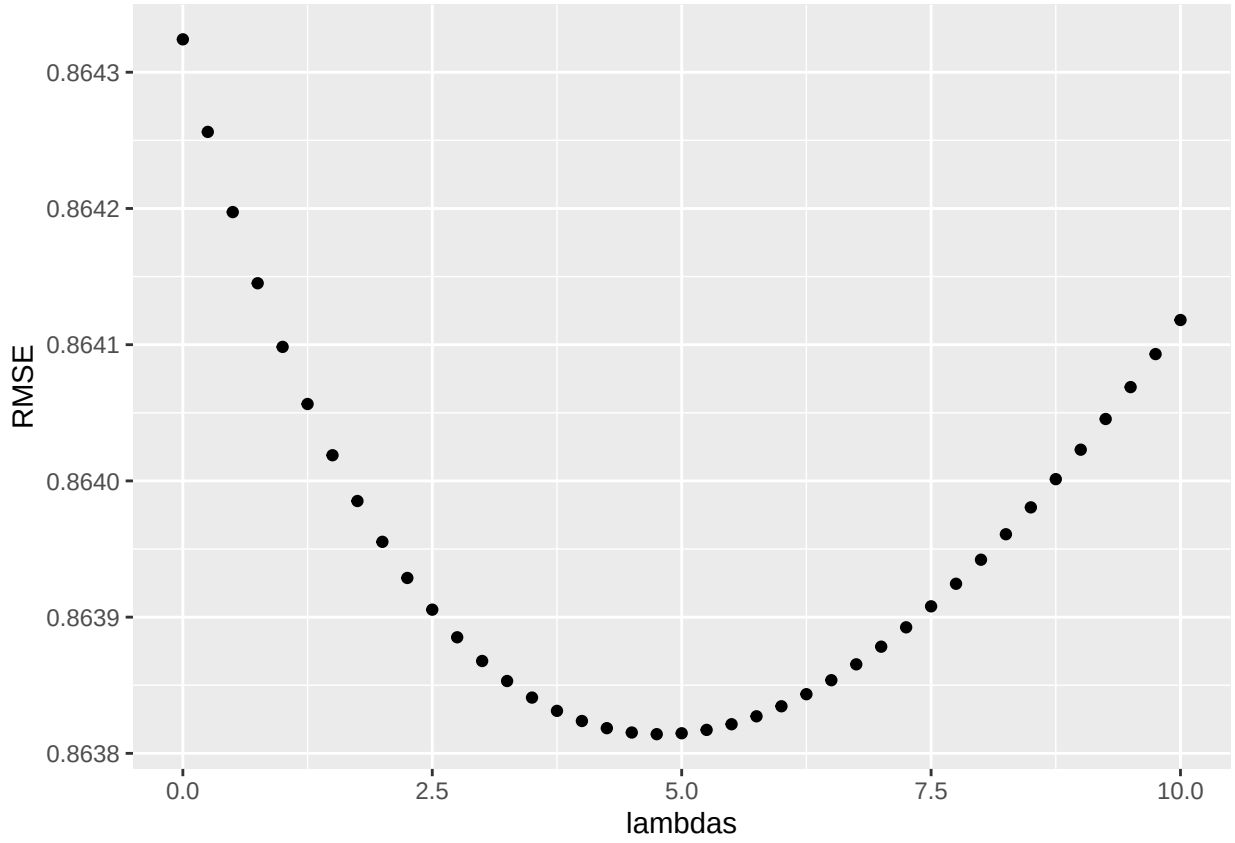
$$1/N \sum_{u,i}(y_{u,i} - \mu - b_i)^2 - \lambda(\sum_i b_i^2 + \sum_u b_u^2 + \sum_g b_g^2)$$

```
mu <- mean(train_set$rating)
lambdas <- seq(0, 10, 0.25)

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

  movie_effects_reg <- train_set %>%
    group_by(movieId)%>%
    summarize(b_i= sum(rating-mu)/(n()+l),n_i=n())

  b_i_reg_test <- test_set %>%
    left_join(movie_effects_reg, by="movieId") %>%
    pull(b_i)

  user_effects_reg <- train_set %>%
    left_join(movie_effects_reg, by="movieId")%>%
    group_by(userId)%>%
    summarize(b_u= sum(rating-mu-b_i)/(n()+l) ,n_i=n())
```

```
  b_u_reg_test <- test_set %>%
    left_join(user_effects_reg, by="userId") %>%
    pull(b_u)


  genres_effects_reg <- train_set %>%
    left_join(movie_effects_reg, by="movieId")%>%
    left_join(user_effects_reg, by="userId")%>%
    group_by(genres)%>%
    summarize(b_g= sum(rating-mu-b_i-b_u)/(n()+l) ,n_i=n())

  b_g_reg_test <-test_set %>%
    left_join(genres_effects_reg, by="genres") %>%
    pull(b_g)


  predicted_ratings <- mu + b_i_reg_test + b_u_reg_test + b_g_reg_test
  RMSE_reg_test <- RMSE(test_set$rating, predicted_ratings)
  return(RMSE_reg_test)

})
```

Now we will use the calculated λ from above to find the RMSE for regularized movie effect and user effect and genre effect model.

```
qplot(lambdas, RMSE)
```




```
lambdas <- lambdas[which.min(RMSE)]
```

```
  movie_effects_reg <- train_set %>%
    group_by(movieId)%>%
    summarize(b_i= sum(rating-mu)/(n()+lambdas),n_i=n())

  b_i_reg_test <- test_set %>%
    left_join(movie_effects_reg, by="movieId") %>%
    pull(b_i)

  user_effects_reg <- train_set %>%
    left_join(movie_effects_reg, by="movieId")%>%
    group_by(userId)%>%
    summarize(b_u= sum(rating-mu-b_i)/(n()+lambdas) ,n_i=n())

  b_u_reg_test <- test_set %>%
    left_join(user_effects_reg, by="userId") %>%
    pull(b_u)


  genres_effects_reg <- train_set %>%
    left_join(movie_effects_reg, by="movieId")%>%
    left_join(user_effects_reg, by="userId")%>%
    group_by(genres)%>%
```

```
    summarize(b_g= sum(rating-mu-b_i-b_u)/(n()+lambdas) ,n_i=n())

  b_g_reg_test <-test_set %>%
    left_join(genres_effects_reg, by="genres") %>%
    pull(b_g)


  predicted_ratings <- mu + b_i_reg_test + b_u_reg_test + b_g_reg_test
  RMSE_reg_test <- RMSE(test_set$rating, predicted_ratings)




RMSE_results <- bind_rows(RMSE_results,
                          data_frame(method="Regularized Movie Effect and User Effect and Genres Effect

RMSE_results %>% knitr::kable()
```

method	RMSE
Average of Rating	1.0600537
Movie Effect Model	0.9429615
Movie Effect and User Effect Model	0.8646843
Movie Effect and User Effect and Genres Effect Model	0.8643241
Regularized Movie Effect Model	0.9429370
Regularized Movie Effect and User Effect Model	0.8641362
Regularized Movie Effect and User Effect and Genres Effect Model	0.8638141

Now we choose the most accurate model as our final model and we can use the validation set to test the final algorithm.

```
#Final RMSE

test_set <- validation

mu <- mean(train_set$rating)
movie_effects_reg <- train_set %>%
  group_by(movieId)%>%
  summarize(b_i= sum(rating-mu)/(n()+lambdas),n_i=n())

b_i_reg_test <- test_set %>%
  left_join(movie_effects_reg, by="movieId") %>%
  pull(b_i)

user_effects_reg <- train_set %>%
  left_join(movie_effects_reg, by="movieId")%>%
  group_by(userId)%>%
  summarize(b_u= sum(rating-mu-b_i)/(n()+lambdas) ,n_i=n())

b_u_reg_test <- test_set %>%
  left_join(user_effects_reg, by="userId") %>%
  pull(b_u)
```

```
genres_effects_reg <- train_set %>%
  left_join(movie_effects_reg, by="movieId")%>%
  left_join(user_effects_reg, by="userId")%>%
  group_by(genres)%>%
  summarize(b_g= sum(rating-mu-b_i-b_u)/(n()+lambdas) ,n_i=n())

b_g_reg_test <-test_set %>%
  left_join(genres_effects_reg, by="genres") %>%
  pull(b_g)


predicted_ratings <- mu + b_i_reg_test + b_u_reg_test + b_g_reg_test
RMSE_reg_test <- RMSE(test_set$rating, predicted_ratings)




RMSE_results <- bind_rows(RMSE_results,
                          data_frame(method="Final RMSE",RMSE = RMSE_reg_test ))

RMSE_results %>% knitr::kable()
```

method	RMSE
Average of Rating	1.0600537
Movie Effect Model	0.9429615
Movie Effect and User Effect Model	0.8646843
Movie Effect and User Effect and Genres Effect Model	0.8643241
Regularized Movie Effect Model	0.9429370
Regularized Movie Effect and User Effect Model	0.8641362
Regularized Movie Effect and User Effect and Genres Effect Model	0.8638141
Final RMSE	0.8648545

Result

Comparing the RMSE result shows that combine movie effect and user effect decreased the RMSE about 8.3% with respect to only movie effect and it is a huge improvement ,and adding genres effect improve it only 0.042%.However, We can decrease the RMSE by regularizing the “Movie Effect and User Effect and Genres Effect Model” about 0.059% and find RMSE equal to 0.8638141 for “Regularized Movie Effect and User Effect and Genres Effect Model”.
We can use the validation set and achieve the final RMSE which is about 0.8648545.