# MovieLens.Recommendation systems project

Hussin Almustafa

2023-08-13

## Introduction:

a Recommender system can be found in almost every information-intensive website. For example, a list of likely preferred products are recommended to an customer when browsing the target product in Amazon , when watching a video clip in Youtube, a recommender system employed in the system suggests some relevant videos to users by learning the users' behaviours that were generated previously. Here we provide the basics of how these recommendations are predicted,motivated by some of the approaches taken by the winners of the Netflix challenge.

here is a link to my github repo

### Methods :

after downloading the data i,m going to explore it, extract the Information from it ,then split the data into training and test sets to test my algorithm. then i will use the data to build and train a machine learning algorithm , using the inputs in one subset to predict Movie ratings in the validation set,by building a several models and comparing them using root mean squared error RMSA as loss function.

**loding data :**

```
############################################################
# Create edx and final_holdout_test sets
############################################################

# Note: this process could take a couple of minutes

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

**Create Train and Final Hold-out Sets:**

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

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
```

```
## v purrr     1.0.1
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
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
```

```
library(tidyverse)
library(caret)

# MovieLens 10M dataset:
# https://grouplens.org/datasets/movielens/10m/
# http://files.grouplens.org/datasets/movielens/ml-10m.zip

options(timeout = 120)

dl <- "ml-10M100K.zip"
if(!file.exists(dl))
  download.file("https://files.grouplens.org/datasets/movielens/ml-10m.zip", dl)

ratings_file <- "ml-10M100K/ratings.dat"
if(!file.exists(ratings_file))
  unzip(dl, ratings_file)

movies_file <- "ml-10M100K/movies.dat"
if(!file.exists(movies_file))
  unzip(dl, movies_file)

ratings <- as.data.frame(str_split(read_lines(ratings_file), fixed("::"), simplify = TRUE),
                         stringsAsFactors = FALSE)
colnames(ratings) <- c("userId", "movieId", "rating", "timestamp")
ratings <- ratings %>%
  mutate(userId = as.integer(userId),
         movieId = as.integer(movieId),
         rating = as.numeric(rating),
         timestamp = as.integer(timestamp))

movies <- as.data.frame(str_split(read_lines(movies_file), fixed("::"), simplify = TRUE),
                        stringsAsFactors = FALSE)
colnames(movies) <- c("movieId", "title", "genres")
movies <- movies %>%
  mutate(movieId = as.integer(movieId))
```

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

# Final hold-out test set will be 10% of MovieLens data
set.seed(1, sample.kind="Rounding") # if using R 3.6 or later
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
# set.seed(1) # if using R 3.5 or earlier
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 final hold-out test set are also in #edx set
final_holdout_test <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from final hold-out test set back into edx set
removed <- anti_join(temp, final_holdout_test)
```

```
## Joining with `by = join_by(userId, movieId, rating, timestamp, title, genres)`
```

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

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

**Data exploration :**

```r
head(edx)
```

```
##   userId movieId rating timestamp                         title
## 1      1     122      5 838985046              Boomerang (1992)
## 2      1     185      5 838983525               Net, The (1995)
## 4      1     292      5 838983421               Outbreak (1995)
## 5      1     316      5 838983392               Stargate (1994)
## 6      1     329      5 838983392 Star Trek: Generations (1994)
## 7      1     355      5 838984474       Flintstones, The (1994)
##                          genres
## 1                Comedy|Romance
## 2          Action|Crime|Thriller
## 4  Action|Drama|Sci-Fi|Thriller
## 5        Action|Adventure|Sci-Fi
## 6 Action|Adventure|Drama|Sci-Fi
## 7       Children|Comedy|Fantasy
```

```r
str(edx)
```

```
## 'data.frame':    9000055 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : int  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|Ac
```

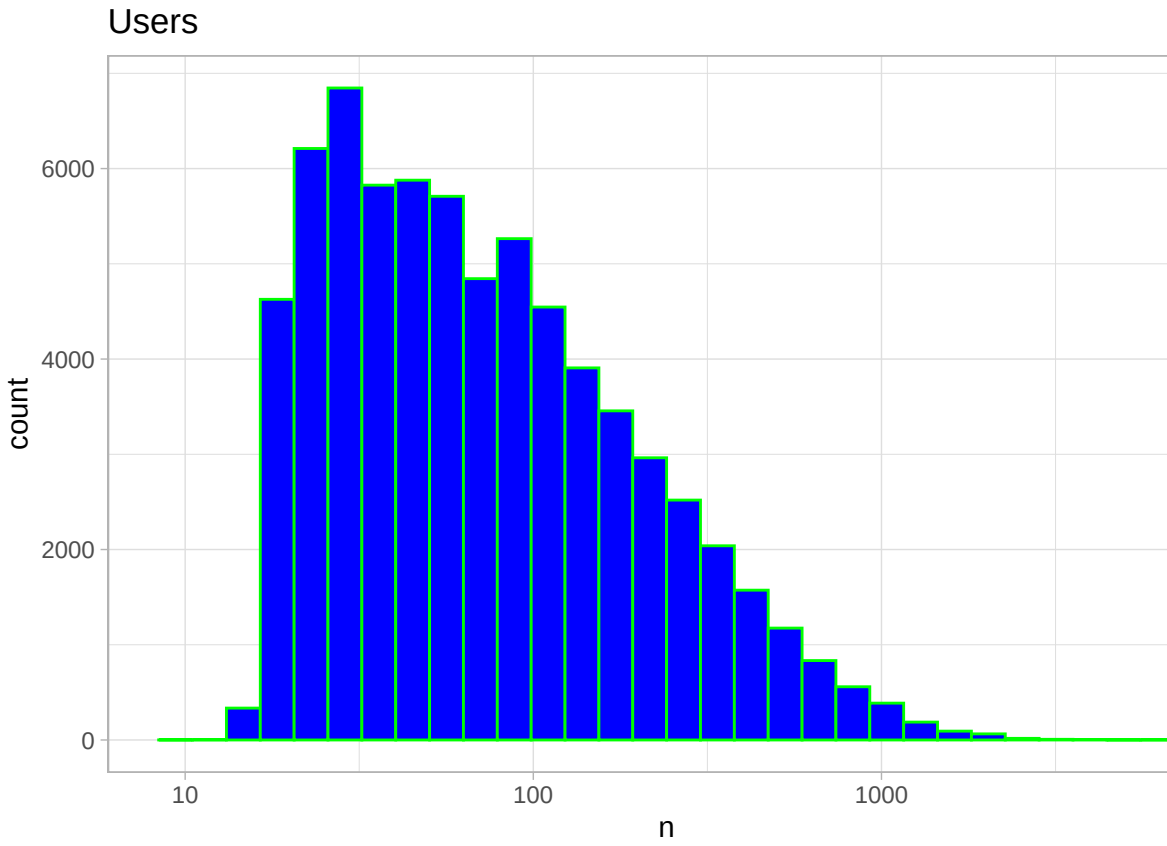
data.frame: 9000055 obs. of 6 variables:

We can see the number of unique users that provide ratings and for how many unique movies they provided them and how many genres

```
 edx %>% summarize(n_users = n_distinct(userId),
                   n_movies = n_distinct(movieId),
                   n_genres = n_distinct(genres))
```

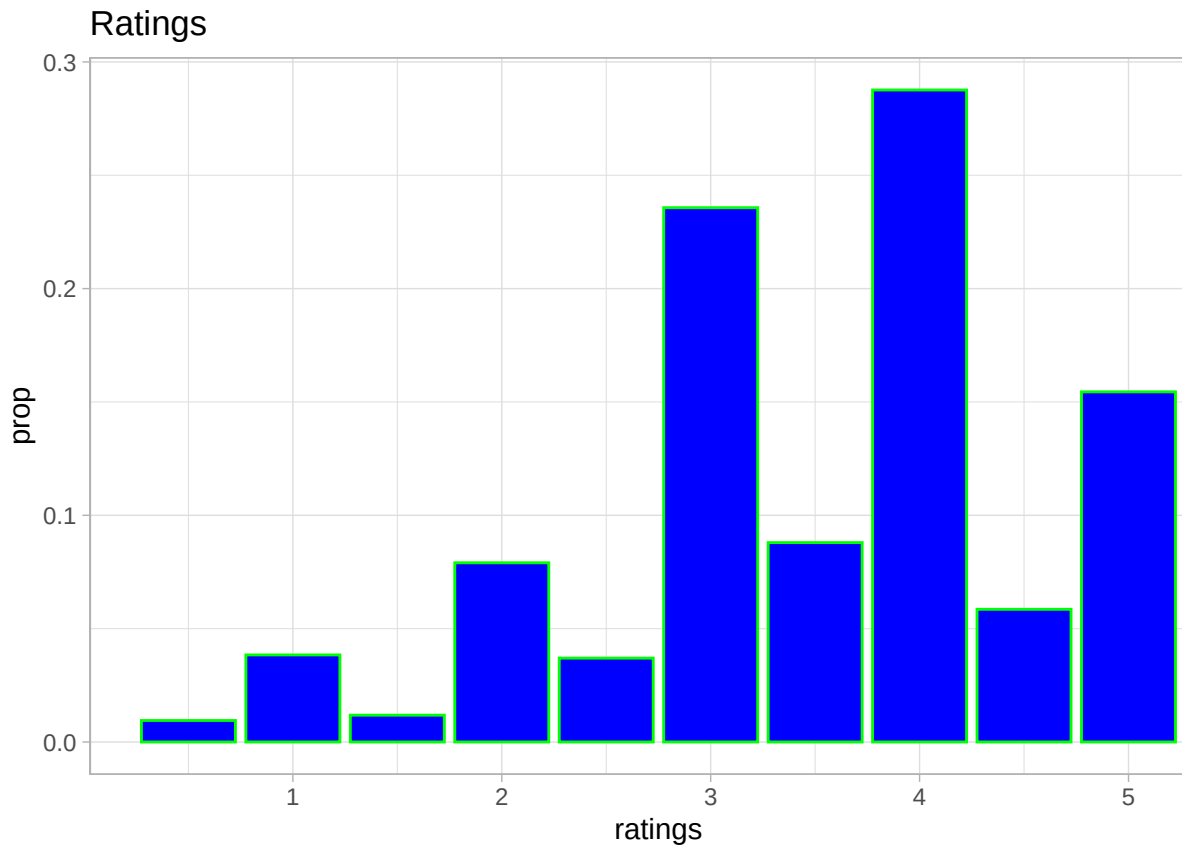
```
##   n_users n_movies n_genres
## 1   69878    10677      797
```

```
edx %>%
  dplyr::count(userId) %>%
  ggplot(aes(n)) +
  geom_histogram( bins= 30 ,color = "green",fill="blue") +
  scale_x_log10() +
  ggtitle("Users")+
  theme_light()
```



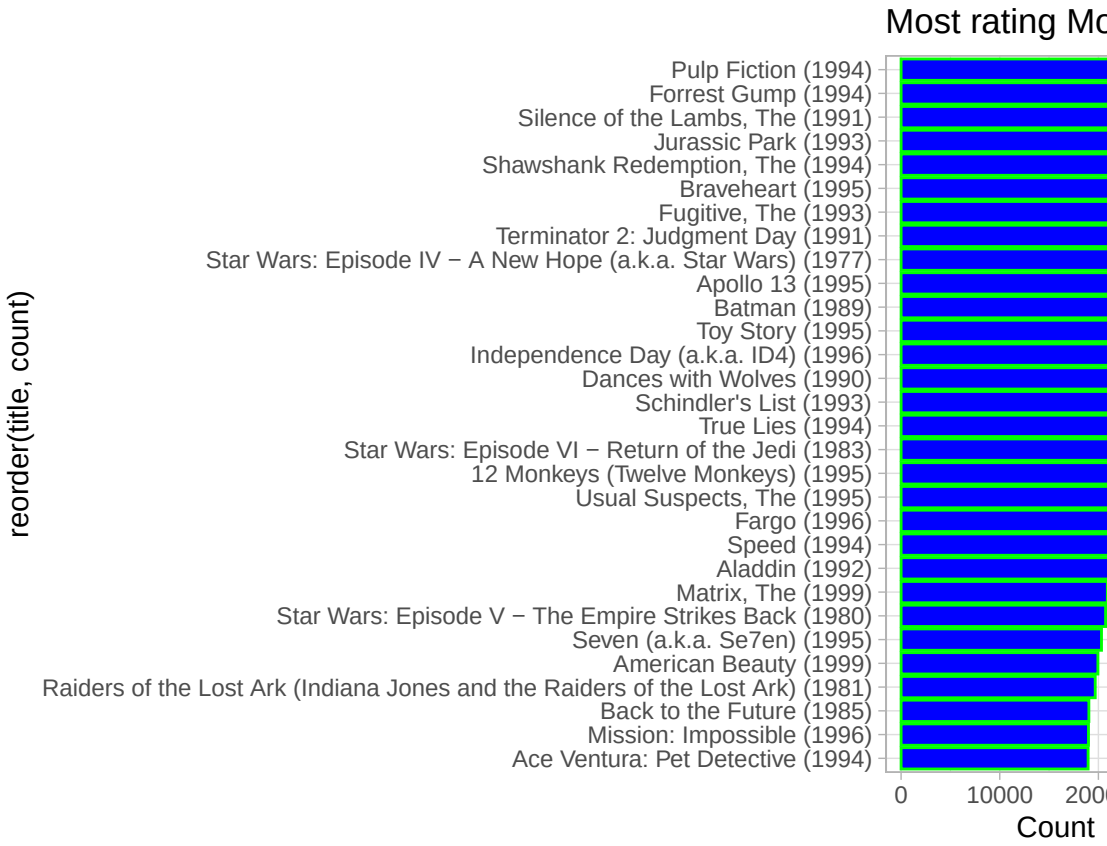

**movies distrbution:**

```
edx %>%
 ggplot(aes(rating, y = after_stat(prop))) +

 geom_bar(color = "green", fill = "blue") +
 labs(x = "ratings" ) +
ggtitle("Ratings")+
 theme_light()
```



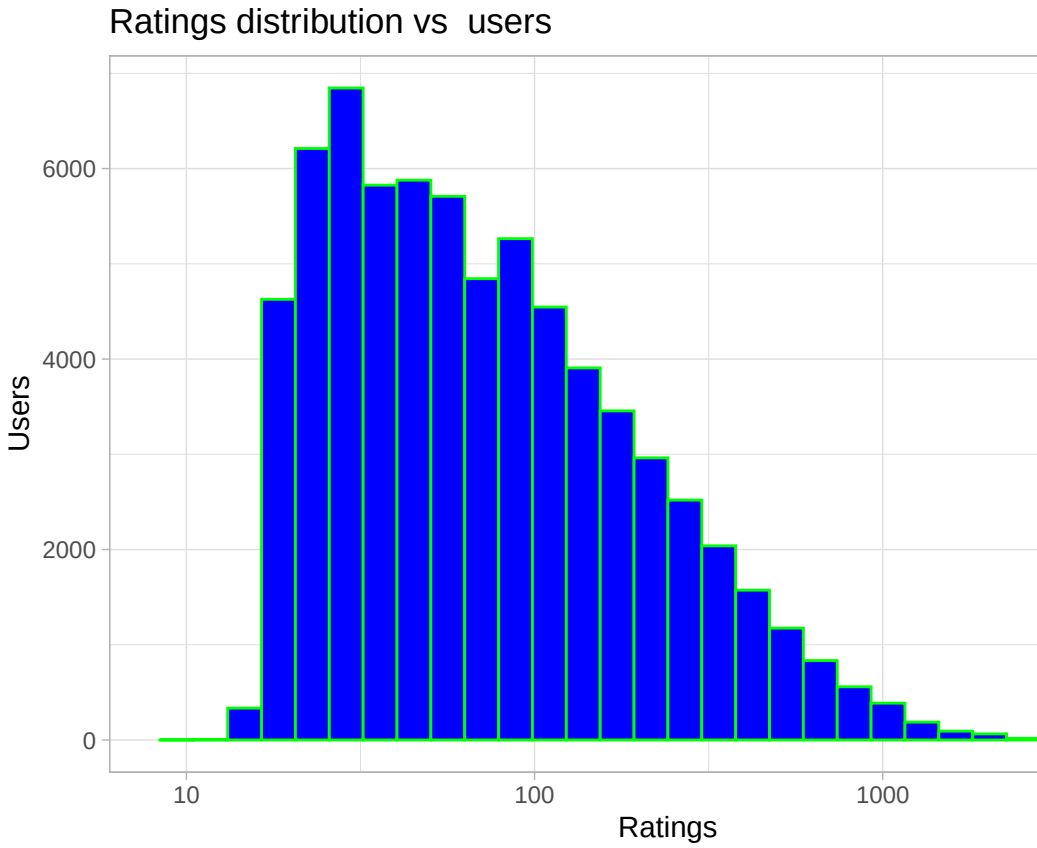

**rating distrbution:**

```
edx %>%
 group_by(title) %>%
 summarize(count = n()) %>%
 arrange(-count) %>%
 top_n(30, count) %>%
 ggplot(aes(count, reorder(title, count))) +
 geom_bar(color ="green", fill = "blue", stat = "identity") +
 xlab("Count") +
 ggtitle("Most rating Movies")+
 theme_light()
```




**most rating movies are :**

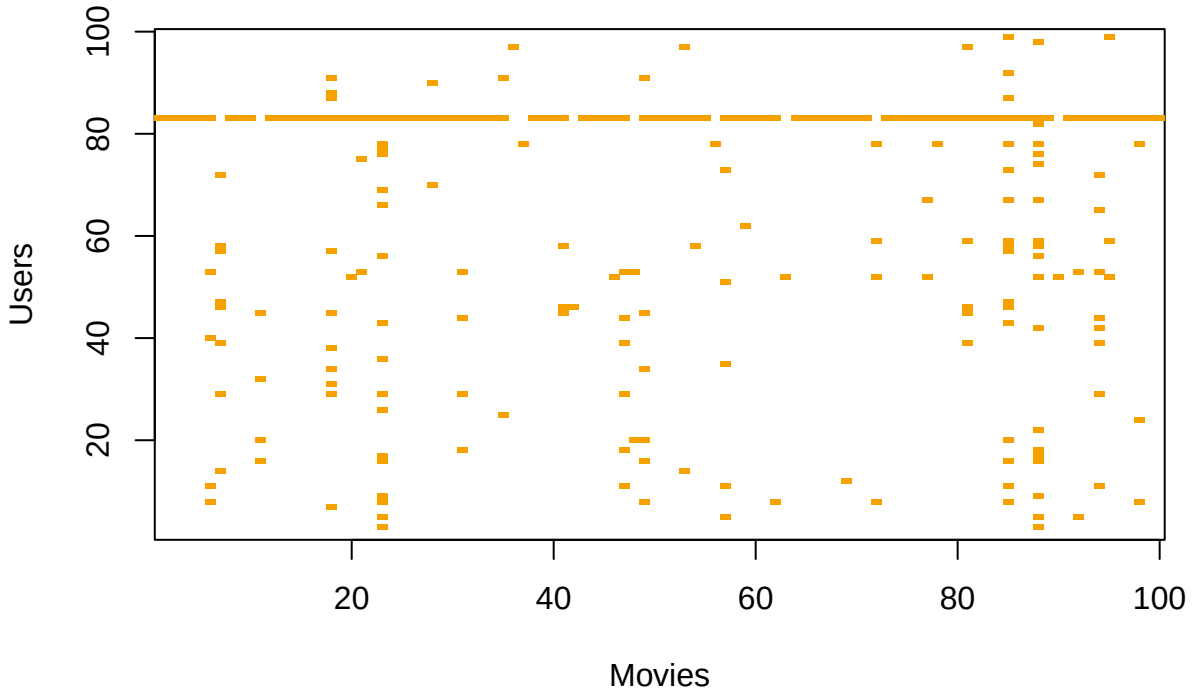
```
edx %>% group_by(userId) %>%
  summarize(count = n()) %>%
  ggplot(aes(count)) +
  geom_histogram(bins=30 ,color = "green", fill = "blue") +
  xlab("Ratings") +
  ylab("Users") +
  scale_x_log10() +
  ggtitle("Ratings distribution vs  users")+
  theme_light()
```




**Rating distribution by users :**

*here is the matrix for a random sample of 100 movies and 100 users:*

```
users <- sample(unique(edx$userId), 100)
 edx %>% filter(userId %in% users) %>%
  select(userId, movieId, rating) %>%
  mutate(rating = 1) %>%
  spread(movieId, rating) %>% select(sample(ncol(.), 100)) %>%
  as.matrix() %>% t(.) %>%
  image(1:100, 1:100,. , xlab="Movies", ylab="Users")
```




```
set.seed(1, sample.kind="Rounding")
```

***use edx to create a test set using caret package :***

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
 #edx_test will be 20 % of edx  data

 test_index <- createDataPartition(y = edx$rating, times = 1,
                                   p = 0.2, list = FALSE)
 edx_train <- edx[-test_index,]
 temp <- edx[test_index,]

##To make sure we don't include users and movies in the test set that do not
# appear in the training set, we removed these using the semi_join function

 edx_test <- temp %>%
   semi_join(edx_train, by = "movieId") %>%
   semi_join(edx_train, by = "userId")

 # Adding back rows into edx_train set
 removed <- anti_join(temp, edx_test)
```

```
## Joining with `by = join_by(userId, movieId, rating, timestamp, title, genres)`
```

```
## Joining with `by = join_by(userId, movieId, rating, timestamp, title, genres)`
 edx_train <- rbind(edx_train, removed)

 rm( temp, removed)
```

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

***To compare different models we will use root mean squared error (RMSE) as our loss function:***

## Building the Recommendation System:

**model-1 Just the average** the simplest possible recommendation system. We're going to predict the same rating for all movies, regardless of the user and movie.

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

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

```
## [1] 3.512478
```

```
#compute the residual mean squared error on the edx_test set

 naive_rmse <- RMSE(edx_test$rating, mu_hat)
 naive_rmse
```

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

```
 rmse_results <- tibble(method = "Just the avg", RMSE = naive_rmse)
rmse_results %>% knitr::kable()
```

***create a table that's going to store the results that we obtain as we go along. We're going to call it rmse_results.***

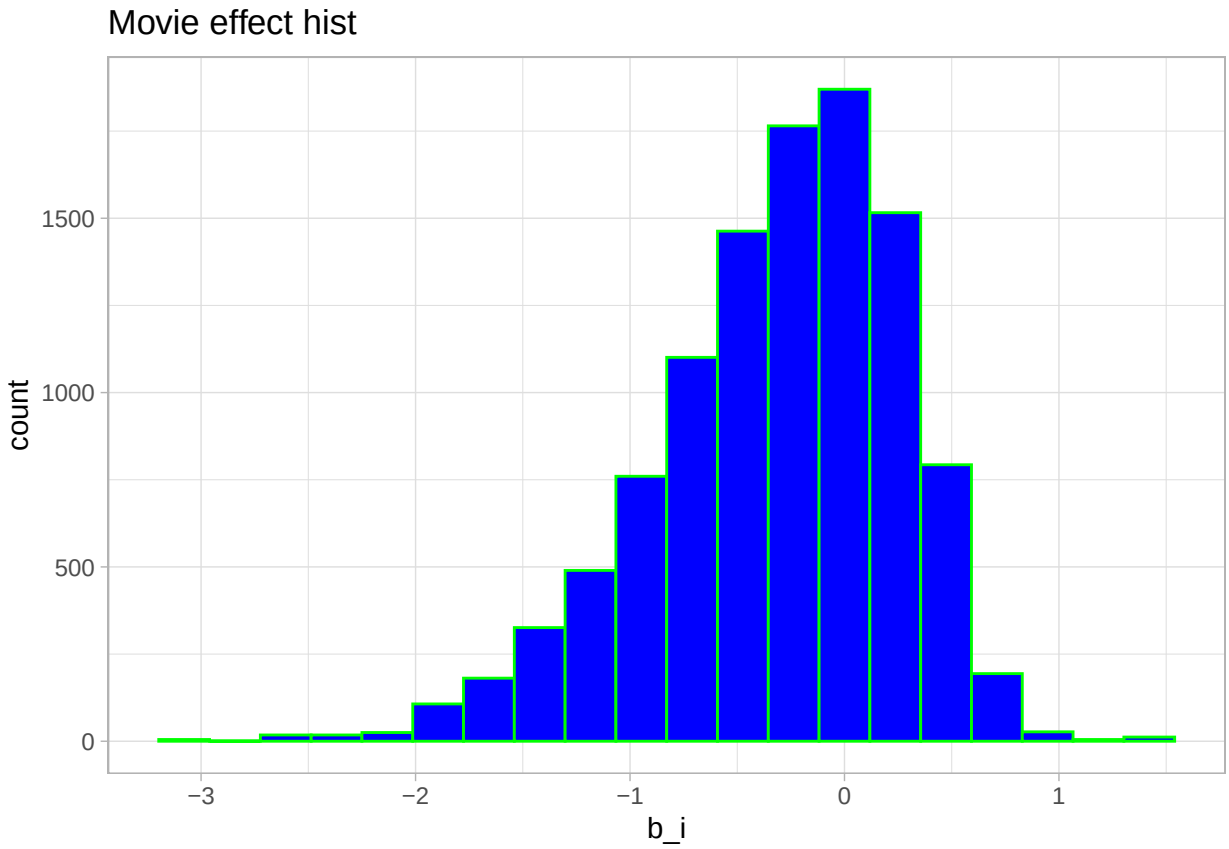
| method | RMSE |
|---|---|
| Just the avg | 1.059904 |

**model-2 Movie Effect:** We can improve our model by adding a term b_i that represents the average rating for movie i

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

```
bi <- edx_train %>%
 group_by(movieId) %>%
 summarize(b_i = mean(rating - mu_hat))
```

```
bi %>% ggplot(aes(b_i)) +
 geom_histogram( bins= 20,color="green",fill="blue")+
 ggtitle("Movie effect hist")+
 theme_light()
```




```
predicted_ratings <- mu_hat + edx_test %>%
 left_join(bi, by="movieId") %>%
 pull(b_i)
```

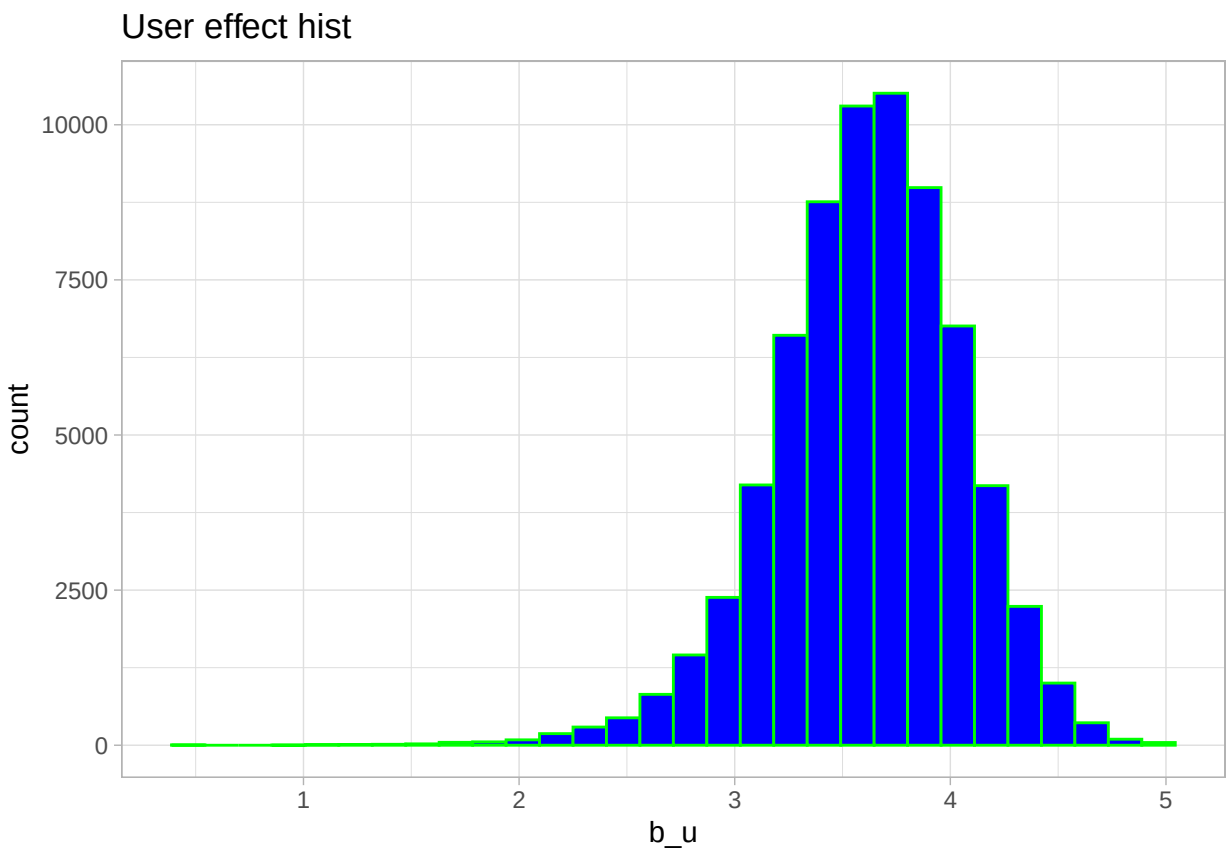
```
 movie_rmse <- RMSE(predicted_ratings, edx_test$rating)
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Avg+Movie Effect",
                                 RMSE = movie_rmse ))

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

| method | RMSE |
|---|---|
| Just the avg | 1.0599043 |
| Avg+Movie Effect | 0.9437429 |

**model_3 user effect:** because different users different in terms of how they rate movies We can further improve our model by adding b_u , the user-specific effect:

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

```
edx_train %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating)) %>%
  filter(n()>=100) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30 ,color = "green", fill="blue")+
  ggtitle("User effect hist")+
  theme_light()
```




```
bu <-edx_train %>%
  left_join(bi, by = "movieId") %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu_hat - b_i))


predicted_ratings <- edx_test %>%
  left_join(bi, by = "movieId") %>%
  left_join(bu, by = "userId") %>%
  mutate(pred = mu_hat + b_i + b_u) %>%
  pull(pred)
```

```
user_rmse <- RMSE(predicted_ratings, edx_test$rating)
rmse_results <- bind_rows(rmse_results,
                tibble(method = "Avg+Movie+user " ,
                                  RMSE = user_rmse))
rmse_results %>% knitr::kable()
```

| method | RMSE |
|---|---|
| Just the avg | 1.0599043 |
| Avg+Movie Effect | 0.9437429 |
| Avg+Movie+user | 0.8659319 |

**model 4 Regularization :** regularization permits us to penalize large estimates that come from small sample sizes. by add a penalty for large values of b to the sum of squares equations that we minimize.

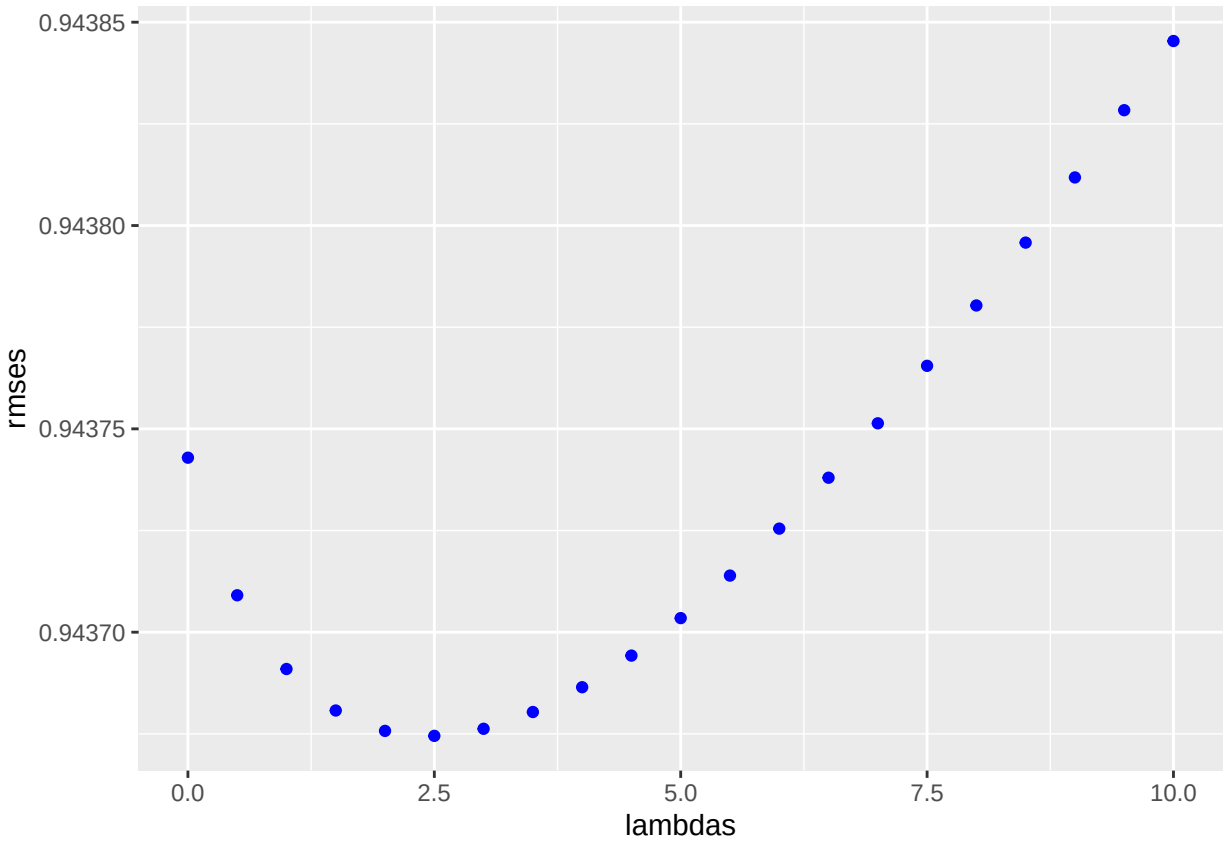
$$\hat{b}_i(\lambda) = \frac{1}{\lambda + n_i} \sum_{u=1}^{n_i} (Y_{u,i} - \hat{\mu})$$

lambda is a tuning parameter.We can use cross-validation to choose it.

```
lambdas <- seq(0, 10, 0.5)
mu <- mean(edx_train$rating)
just_the_sum <- edx_train %>%
  group_by(movieId) %>%
  summarize(s = sum(rating - mu), n_i = n())
rmses <- sapply(lambdas, function(l){
  predicted_ratings <- edx_test %>%
    left_join(just_the_sum, by='movieId') %>%
    mutate(b_i = s/(n_i+l)) %>%
    mutate(pred = mu + b_i) %>%
    .$pred
  return(RMSE(predicted_ratings, edx_test$rating))
})
```

```
qplot(lambdas, rmses,color=I("blue"))
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

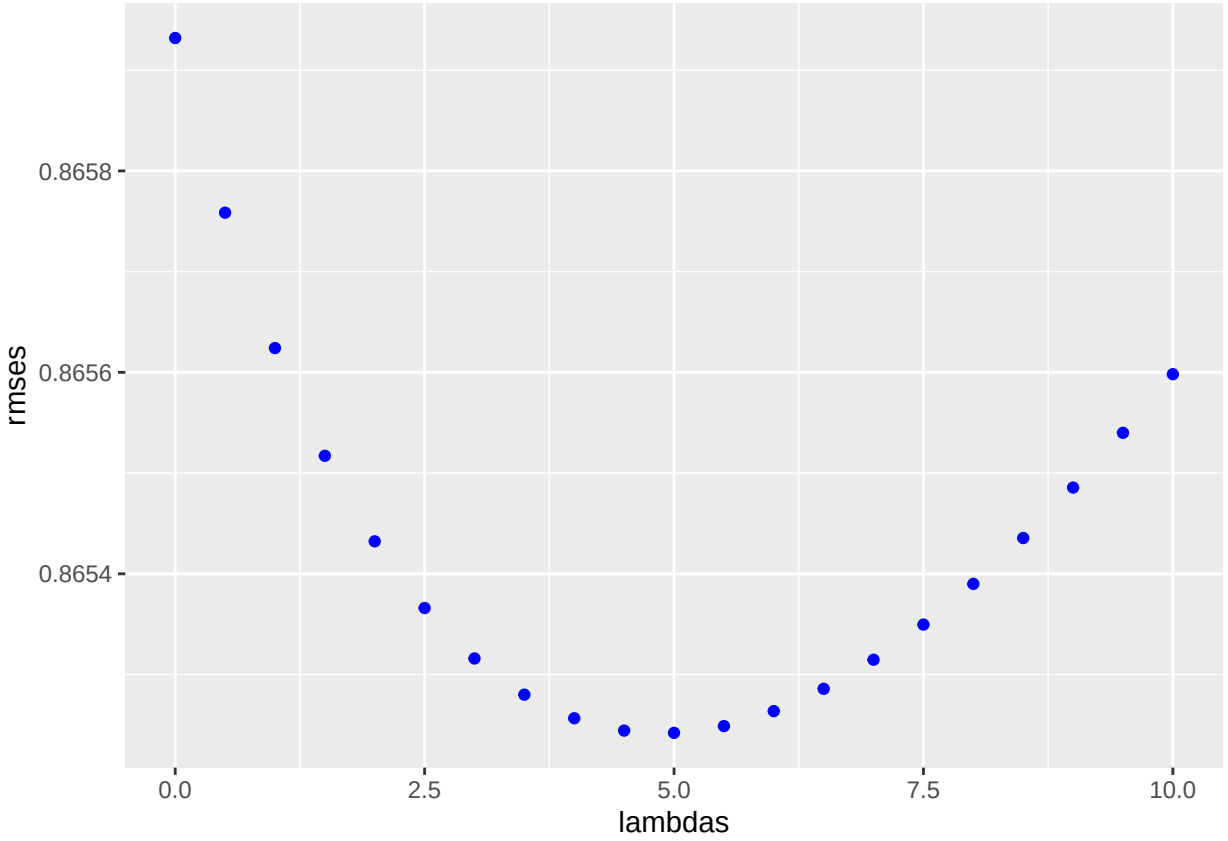



```
lambdas[which.min(rmses)]
```

```
## [1] 2.5
```

We can also use regularization to estimate the user effect

```
lambdas <- seq(0, 10, 0.5)
rmses <- sapply(lambdas, function(x){
  mu <- mean(edx_train$rating)
  b_i <- edx_train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+x))
  b_u <- edx_train %>%
    left_join(b_i, by= "movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+x))
  predicted_ratings <- edx_test %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    pull(pred)
  return(RMSE(predicted_ratings, edx_test$rating))
})

qplot(lambdas, rmses, color=I("blue"))
```




```
lambda <- lambdas[which.min(rmses)]
lambda
```

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

```
rmse_results <- bind_rows(rmse_results,
                          tibble(method="Regularized Movie  User ",
                                 RMSE = min(rmses)))
rmse_results %>% knitr::kable()
```

| method | RMSE |
|---|---|
| Just the avg | 1.0599043 |
| Avg+Movie Effect | 0.9437429 |
| Avg+Movie+user | 0.8659319 |
| Regularized Movie User | 0.8652421 |

```
# Create a model object  by calling Reco()
library(recosystem)
set.seed(111, sample.kind="Rounding")
```

**model 5 matrix factorization:**

```
## Warning in set.seed(111, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
edx_train_reco <- with(edx_train, data_memory(user_index =
                         userId, item_index = movieId, rating = rating ,
                         package = "recosystem" ))
edx_test_reco <- with(edx_test, data_memory(user_index = userId,
                   item_index = movieId, rating = rating,
                   package = "recosystem"))


r <- Reco()



p_reco <- r$tune(edx_train_reco, opts = list(dim = c(10,20, 30),
                                             costp_l2 = c(0.1, 0.5),
                                             costq_l2 = c(0.1, 0.5),
                                             lrate = c(0.1, 0.5),
                                             nthread = 4,
                                             niter = 10))
# Train the model by calling the train()
r$train(edx_train_reco, opts = c(p_reco$min, nthread = 4, niter = 30))
```

```
## iter      tr_rmse          obj
##    0       0.9898   1.2268e+07
##    1       0.8855   1.0730e+07
##    2       0.8633   1.0551e+07
##    3       0.8480   1.0394e+07
##    4       0.8400   1.0334e+07
##    5       0.8327   1.0289e+07
##    6       0.8269   1.0249e+07
##    7       0.8225   1.0218e+07
##    8       0.8188   1.0198e+07
##    9       0.8156   1.0182e+07
##   10       0.8129   1.0170e+07
##   11       0.8105   1.0157e+07
##   12       0.8084   1.0145e+07
##   13       0.8066   1.0138e+07
##   14       0.8050   1.0129e+07
##   15       0.8036   1.0125e+07
##   16       0.8021   1.0116e+07
##   17       0.8009   1.0113e+07
##   18       0.7998   1.0106e+07
##   19       0.7988   1.0101e+07
##   20       0.7978   1.0099e+07
##   21       0.7969   1.0095e+07
##   22       0.7959   1.0090e+07
##   23       0.7952   1.0084e+07
##   24       0.7946   1.0085e+07
##   25       0.7938   1.0082e+07
##   26       0.7932   1.0079e+07
##   27       0.7926   1.0076e+07
##   28       0.7920   1.0073e+07
```

```
##   29       0.7915   1.0074e+07
```

```
results_reco <- r$predict(edx_test_reco, out_memory())

# our RMSE for this model

reco_rmse <- RMSE(results_reco, edx_test$rating)
rmse_results  <- bind_rows(rmse_results , tibble(method =
                          "matrix factorization ", RMSE = reco_rmse))

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

| method | RMSE |
|---|---|
| Just the avg | 1.0599043 |
| Avg+Movie Effect | 0.9437429 |
| Avg+Movie+user | 0.8659319 |
| Regularized Movie User | 0.8652421 |
| matrix factorization | 0.8174689 |

```
set.seed(111, sample.kind="Rounding")
```

**applying model 5 on final_holdout_test set :**

```
## Warning in set.seed(111, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
edx_reco <- with(edx, data_memory(user_index =
                 userId, item_index = movieId, rating = rating ,
                 package = "recosystem" ))


final_holdout_test_reco <- with( final_holdout_test,
                             data_memory(user_index =userId,
                             item_index = movieId, rating = rating,
                              package = "recosystem"))



r <- Reco()


p_final_reco <- r$tune(edx_reco, opts = list(dim = c(10,20, 30),
                                             costp_l2 = c(0.1, 0.5),
                                             costq_l2 = c(0.1, 0.5),
                                             lrate = c(0.1, 0.5),
                                             nthread = 4,
                                             niter = 10))
# Train the  model  on edx set by calling the train()
r$train(edx_reco, opts = c(p_final_reco$min, nthread = 4, niter = 30))
```

```
## iter      tr_rmse          obj
##    0       0.9708   1.4961e+07
##    1       0.8803   1.3399e+07
##    2       0.8544   1.3072e+07
##    3       0.8437   1.2947e+07
##    4       0.8358   1.2888e+07
##    5       0.8292   1.2834e+07
##    6       0.8243   1.2792e+07
##    7       0.8204   1.2772e+07
##    8       0.8173   1.2748e+07
##    9       0.8145   1.2728e+07
##   10       0.8125   1.2718e+07
##   11       0.8107   1.2707e+07
##   12       0.8091   1.2694e+07
##   13       0.8077   1.2685e+07
##   14       0.8065   1.2681e+07
##   15       0.8054   1.2673e+07
##   16       0.8043   1.2664e+07
##   17       0.8034   1.2663e+07
##   18       0.8026   1.2658e+07
##   19       0.8018   1.2651e+07
##   20       0.8012   1.2650e+07
##   21       0.8004   1.2643e+07
##   22       0.7997   1.2638e+07
##   23       0.7992   1.2637e+07
##   24       0.7986   1.2633e+07
##   25       0.7981   1.2634e+07
##   26       0.7976   1.2632e+07
##   27       0.7972   1.2627e+07
##   28       0.7968   1.2624e+07
##   29       0.7964   1.2623e+07
```

```
final_results_reco <- r$predict(final_holdout_test_reco, out_memory())
```

```
final_reco_rmse <- RMSE(final_results_reco, final_holdout_test $rating)
rmse_results  <- bind_rows(rmse_results , tibble(method =
                           "final matrix factorization ",
                              RMSE = final_reco_rmse))


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

**our final RMSE for this model on final_hold_out set:**

| method | RMSE |
|---|---|
| Just the avg | 1.0599043 |
| Avg+Movie Effect | 0.9437429 |
| Avg+Movie+user | 0.8659319 |
| Regularized Movie User | 0.8652421 |

| method | RMSE |
|---|---|
| matrix factorization | 0.8174689 |
| final matrix factorization | 0.8160745 |

**Citations:** Irizarry, Rafael A. -Introduction to Data Science-

Yixuan Qiu - recosystem: Recommender System Using Parallel Matrix Factorization -