MovieLens Project Report

Monica Bustamante

HarvardX-Data Science: Capstone

INTRODUCTION

Movielenss is a project developed by GroupLens, a research laboratory at the University of Minnesota. MovieLens provides online movie recommender algorithms, the full data set consists of more de 25 million ratings across more than 40,000 movies by more than 250.000 users, all users selected had rated at least 20 movies, each user is represented by id. This project will predict features and the rating of movies by users using ratings that have been collected for several years by Movilens and thus convert them to algorithms and machine learning models, and then recommend users in their future searches, as a result, verify the performance of algorithms. For the evaluation, the residual mean square error (RMSE) of the predictions will be used and thus compare the real rating of the users. In general, the algorithms and model will show a deep understanding of the variables, observations, and ratings given by users, and as a result, compare the final results and predictions.

1. Create Edx Set, validation set
2. Install Packages
3. Install Libraries
4. Load Data set from HTTP
5. Create Rating
6. Split data
7. Create DataFrame
8. Create validation set
9. Analysis of the variables
10. Model Developing Approach

1. Installing essential Packages and Libraries

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

```
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
```

```r
#Install packages
install.packages("data.table", repos = "http://cran.us.r-project.org")
```

```
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
```

```r
#Install packages caret
install.packages("caret", repos = "http://cran.us.r-project.org", dependencies=TRUE)
```

```
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
```

```r
#Install library
library(tidyverse)
library(caret)
```

```
Warning message in system("timedatectl", intern = TRUE):
"running command 'timedatectl' had status 1" Attaching packages  tidyverse 1.3.1
 ggplot2 3.3.3      purrr   0.3.4
 tibble  3.1.2      dplyr   1.0.6
 tidyr   1.1.3      stringr 1.4.0
 readr   1.4.0      forcats 0.5.1
 Conflicts  tidyverse_conflicts()
 dplyr::filter()     masks stats::filter()
 dplyr::group_rows() masks kableExtra::group_rows()
 dplyr::lag()        masks stats::lag()
Loading required package: lattice

Attaching package: 'caret'

The following object is masked from 'package:purrr':

    lift
```

2. DataSet Downloading

Data set is from the web http://files.grouplens.org/datasets/movielens/ml-10m.zip", and it is stored in temporary file.

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

```r
#Read table
ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                      col.names = c("userId", "movieId", "rating", "timestamp"))
```

```r
#Split dataset
movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
colnames(movies) <- c("movieId", "title", "genres")
```

```r
#Mutate, rename title
movies <- as.data.frame(movies) %>% mutate(movieId = as.numeric(levels(movieId))[movieId],
                                           title = as.character(title),
                                           genres = as.character(genres))
```

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

###

Split and Validation:

```
Prepared the Movilens dataset split and validation by 10%.
```

```r
# 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 = FALSE)
 edx <- movielens[-test_index,]
 temp <- movielens[test_index,]
```

```r
# 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")
```

```r
# 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)
```

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

```r
#validation dataset
validation  <- validation %>% select(-rating)
```

3. Data Cleansing

Verify nan values in Edx validation dataframes:

```
na_edx <- sapply(edx, function(x) sum(is.na(x)))
na_validation <- sapply(validation, function(x) sum(is.na(x)))
print(na_edx, na_validation)
```

```
   userId   movieId    rating timestamp     title    genres
        0         0         0         0         0         0
```

4. Basic information at Data Set

Acquire information by exploring and analyzing the dataset, understanding the effects of the different variables.

How many rows and columns are there in the edx dataset?

```
#To see more information about the dataset
head(edx, 5)
```

userId

movieId

rating

timestamp

title

genres

1

122

5

838985046

Boomerang (1992)

Comedy|Romance

1

185

5

838983525

Net, The (1995)

Action|Crime|Thriller

1

231

5

838983392

Dumb & Dumber (1994)

Comedy

1

292

5

838983421

Outbreak (1995)

Action|Drama|Sci-Fi|Thriller

1

316

5

838983392

Stargate (1994)

Action|Adventure|Sci-Fi

```
#General information about dataset
summary(edx)
```

```
     userId         movieId          rating        timestamp
 Min.   :    1   Min.   :    1   Min.   :0.500   Min.   :7.897e+08
 1st Qu.:18122   1st Qu.:  648   1st Qu.:3.000   1st Qu.:9.468e+08
 Median :35743   Median : 1834   Median :4.000   Median :1.035e+09
 Mean   :35869   Mean   : 4120   Mean   :3.512   Mean   :1.033e+09
 3rd Qu.:53602   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
 Length:9000061     Length:9000061
 Class :character   Class :character
 Mode  :character   Mode  :character
```

The edx data has 9,000,055 rows or observations and 6 columns or variables. 69,878 users rated one, 797 genres, and more of the 10,677 movies. Each row represents one user's rating to a single movie.

The UserId has Median 35743, while Median is 1834 by MovieId, rating is 4.

```
#How many rows and columns are there in the edx dataset
paste('The edx dataset has',nrow(edx),'rows and',ncol(edx),'columns.')
```

'The edx dataset has 9000061 rows and 6 columns.'

```r
#To see more information about dataset
edx %>% summarise(
  uniq_movies = n_distinct(movieId),
  uniq_users = n_distinct(userId),
  uniq_genres = n_distinct(genres))
```

uniq_movies

uniq_users

uniq_genres

10677

69878

797

```r
#Mean or average of rating dataset
rating_mean <- mean(edx$rating)
rating_mean
```

3.51246397107753

How many zeros were given as ratings in the edx dataset?

```r
#How many zeros were given as ratings in the edx dataset.
paste(sum(edx$rating == 0), 'ratings with 0 were given and',
      sum(edx$rating == 3),'ratings with 3')
```

'0 ratings with 0 were given and 2121638 ratings with 3'

```r
edx %>% filter(rating == 3) %>% tally()
```

n

2121638

How many different movies are in the edx dataset?

```r
#How many different movies are in the edx dataset
n_distinct(edx$movieId)
```

10677

```r
edx %>% summarize(n_movies = n_distinct(movieId))
```

n_movies

10677

How many different users are in the edx dataset?

```r
#How many different users are in the edx dataset. n_distinct or lenght
n_distinct(edx$userId)
```

69878

```r
edx %>% summarize(n_users = n_distinct(userId))
```

n_users

69878

How many movie ratings are in each of the following genres in the edx dataset?

```r
# str_detect
genres = c("Drama", "Comedy", "Thriller", "Romance")
sapply(genres, function(g) {
    sum(str_detect(edx$genres, g))
})

# separate_rows, much slower.
edx %>% separate_rows(genres, sep = "\\|") %>%
    group_by(genres) %>%
    summarize(count = n()) %>%
    arrange(desc(count))
```

Drama

<dd>3909401</dd>
<dt>Comedy</dt>
    <dd>3541284</dd>
<dt>Thriller</dt>
    <dd>2325349</dd>
<dt>Romance</dt>
    <dd>1712232</dd>

genres

count

Drama

3909401

Comedy

3541284

Action

2560649

Thriller

2325349

Adventure

1908692

Romance

1712232

Sci-Fi

1341750

Crime

1326917

Fantasy

925624

Children

737851

Horror

691407

Mystery

567865

War

511330

Animation

467220

Musical

432960

Western

189234

Film-Noir

118394

Documentary

93252

IMAX

8190

(no genres listed)

6

```
#Movie ratings by Drama. str_detect Detect The Presence Or Absence Of A Pattern In A String.
drama <- edx %>% filter(str_detect(genres,"Drama"))
paste('Drama has',nrow(drama),'movies')
```

'Drama has 3909401 movies'

```
#Movie ratings by Comedy
comedy <- edx %>% filter(str_detect(genres,"Comedy"))
paste('Comedy has',nrow(comedy),'movies')
```

'Comedy has 3541284 movies'

```
##Movie ratings by Thriller
thriller <- edx %>% filter(str_detect(genres,"Thriller"))
paste('Thriller has',nrow(thriller),'movies')
```

'Thriller has 2325349 movies'

```
#Movie ratings by Romance
romance <- edx %>% filter(str_detect(genres,"Romance"))
paste('Romance has',nrow(romance),'movies')
```

'Romance has 1712232 movies'

VARIABLE ANALYSIS BY RATING

Find any insights to develop the recommendation model. The qualification is the classification of the information that allows it to be evaluated and valued based on a comparative evaluation of its standard quality or performance, quantity, or its combination. In the Movilens data set, the rating has a numerical ordinal scale of 0.5 to 5 stars from movie viewers. The maximum rating they give 5 stars or less if they do not like the movie.

Which movie has the greatest number of ratings?

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

rating

n

0.5

85420

1.0

345935

1.5

106379

2.0

710998

2.5

332783

3.0

2121638

3.5

792037

4.0

2588021

4.5

526309

5.0

1390541

```
#Greatest number of ratings. Arrange rows by variables
edx %>% group_by(title) %>%
summarise(number = n()) %>%
arrange(desc(number))
```

title

number

Pulp Fiction (1994)

31336

Forrest Gump (1994)

31076

Silence of the Lambs, The (1991)

30280

Jurassic Park (1993)

29291

Shawshank Redemption, The (1994)

27988

Braveheart (1995)

26258

Terminator 2: Judgment Day (1991)

26115

Fugitive, The (1993)

26050

Star Wars: Episode IV - A New Hope (a.k.a. Star Wars) (1977)

25809

Batman (1989)

24343

Apollo 13 (1995)

24277

Toy Story (1995)

23826

Independence Day (a.k.a. ID4) (1996)

23360

Dances with Wolves (1990)

23312

Schindler's List (1993)

23234

True Lies (1994)

22786

Star Wars: Episode VI - Return of the Jedi (1983)

22629

12 Monkeys (Twelve Monkeys) (1995)

21959

Usual Suspects, The (1995)

21533

Speed (1994)

21384

Fargo (1996)

21370

Aladdin (1992)

21214

Matrix, The (1999)

20894

Star Wars: Episode V - The Empire Strikes Back (1980)

20836

Seven (a.k.a. Se7en) (1995)

20271

American Beauty (1999)

19859

Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)

19604

Back to the Future (1985)

19141

Mission: Impossible (1996)

18969

Ace Ventura: Pet Detective (1994)

18907

. . .

. . .

Nazis Strike, The (Why We Fight, 2) (1943)

1

Neil Young: Human Highway (1982)

1

Once in the Life (2000)

1

One Hour with You (1932)

1

Part of the Weekend Never Dies (2008)

1

Please Vote for Me (2007)

1

Prelude to War (Why We Fight, 1) (1943)

1

Prisoner of Paradise (2002)

1

Quiet City (2007)

1

Relative Strangers (2006)

1

Revenge of the Ninja (1983)

1

Ring of Darkness (2004)

1

Rockin' in the Rockies (1945)

1

Säg att du älskar mig (2006)

1

Shadows of Forgotten Ancestors (1964)

1

Splinter (2008)

1

Spooky House (2000)

1

Stacy's Knights (1982)

1

Sun Alley (Sonnenallee) (1999)

1

Symbiopsychotaxiplasm: Take One (1968)

1

Testament of Orpheus, The (Testament d'Orphée) (1960)

1

Thérèse (2004)

1

Tokyo! (2008)

1

Train Ride to Hollywood (1978)

1

Variety Lights (Luci del varietà) (1950)

1

Where A Good Man Goes (Joi gin a long) (1999)

1

Wings of Eagles, The (1957)

1

Women of the Night (Yoru no onnatachi) (1948)

1

Won't Anybody Listen? (2000)

1

Zona Zamfirova (2002)

1

```r
head(sort(-table(edx$rating)), 5)
hist(edx$rating)
summary(edx$rating)
```

```
      4        3        5      3.5        2
-2588021 -2121638 -1390541  -792037  -710998
```

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

```r
edx %>%  # Ratings Distribution:
  ggplot(aes(rating)) +
  geom_histogram(binwidth = 0.15, color = "yellow") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  scale_y_continuous(breaks = c(seq(0, 3000000, 500000))) +
  ggtitle("Graphic Rating Distribution")
```

```
Warning message:
"Continuous limits supplied to discrete scale.
Did you mean `limits = factor(...)` or `scale_*_continuous()`?"
```

What are the five most given ratings in order from most to least?

```r
#Sort a variable in descending order.
edx %>% group_by(rating) %>%
summarize(count = n()) %>%
top_n(5) %>%
arrange(desc(count))
```

```
Selecting by count
```

rating

count

4.0

2588021

3.0

2121638

5.0

1390541

3.5

792037

2.0

710998

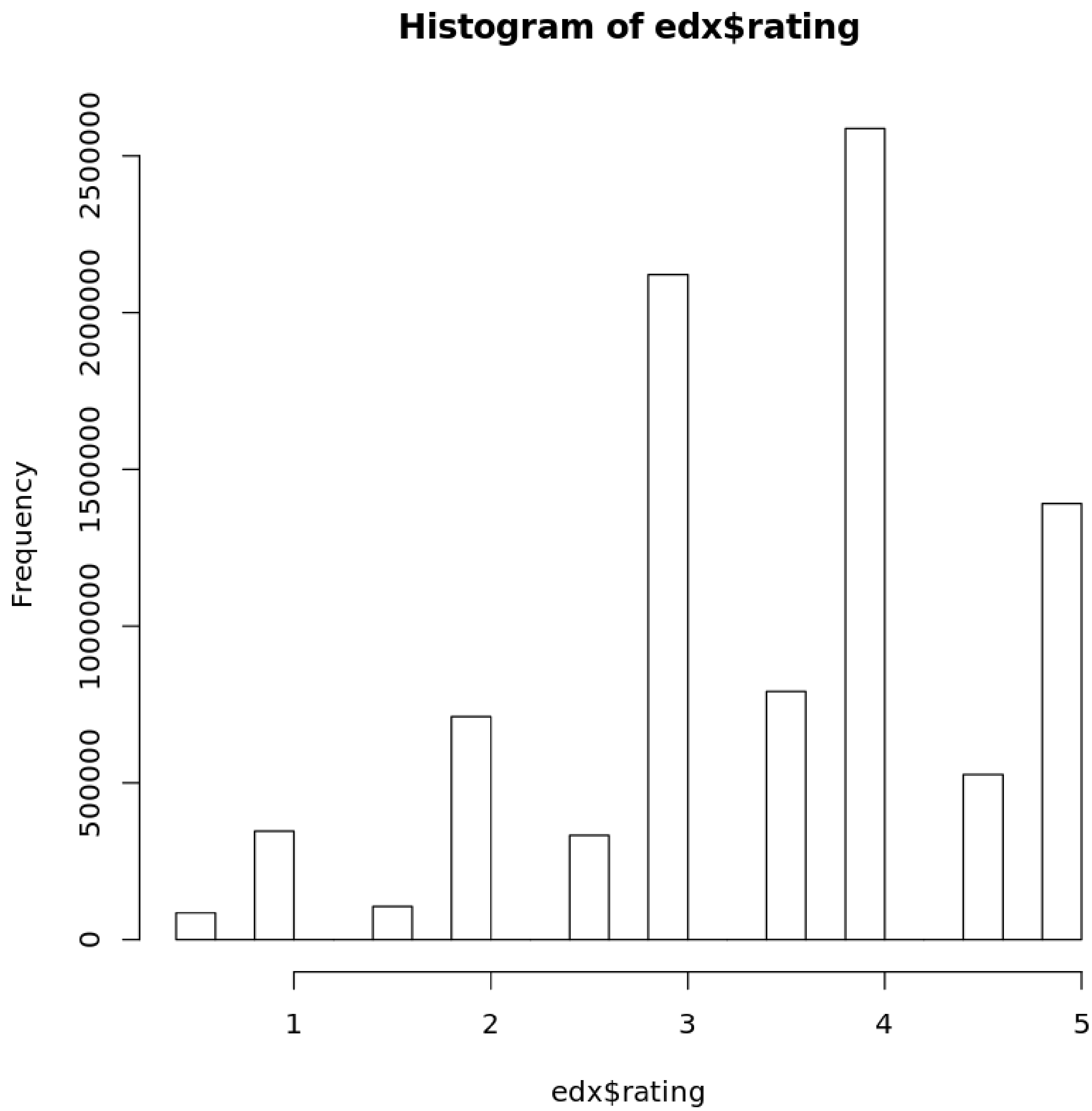

Histogram of edx$rating
Frequency
0
500000
1000000
1500000
2000000
2500000
1
2
3
4
5
edx$rating

Figure 1: png

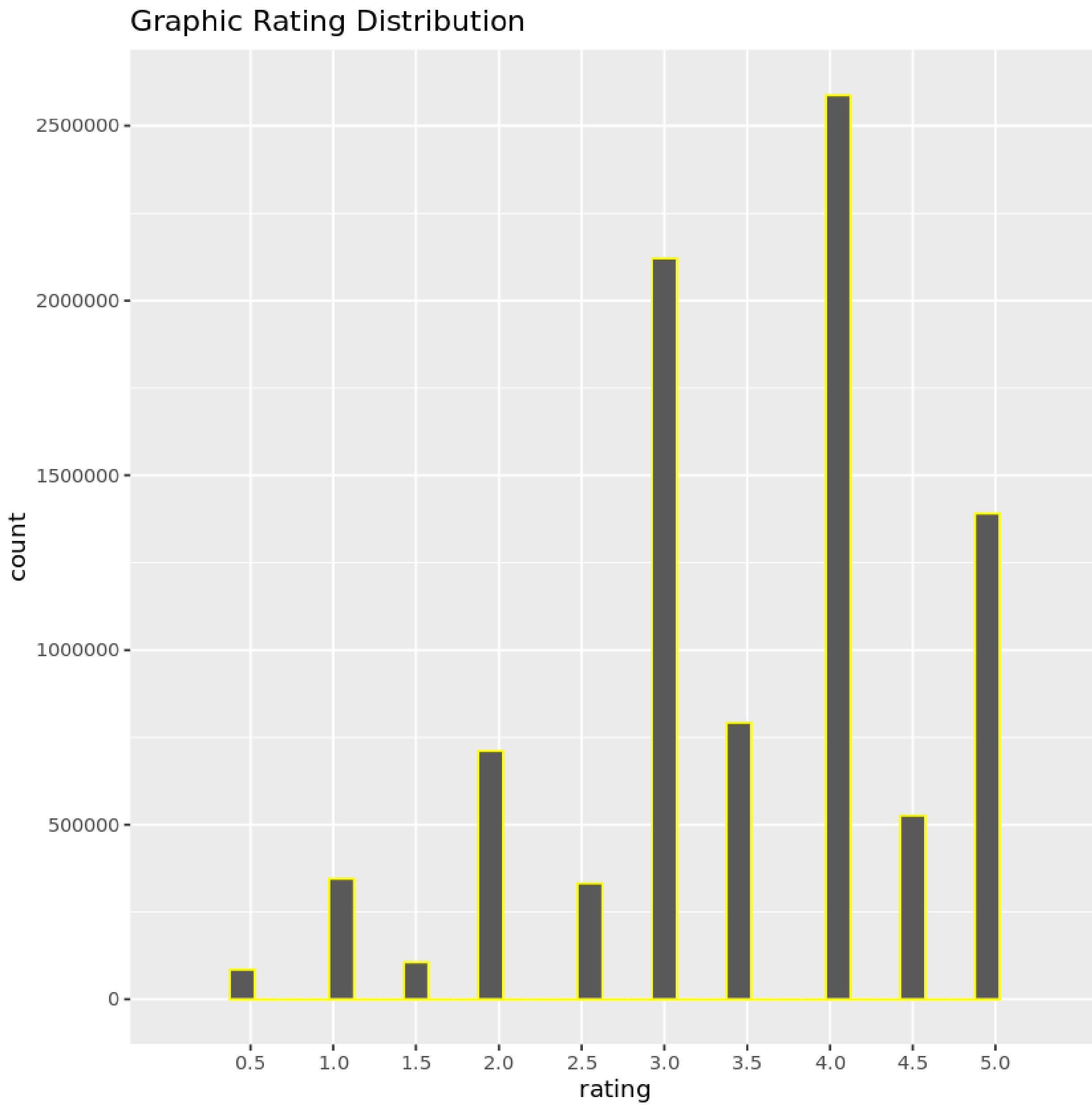

Graphic Rating Distribution
count
2500000
2000000
1500000
1000000
500000
0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
rating

Figure 2: png

True or False: In general, half star ratings are less common than whole star ratings (e.g., there are fewer ratings of 3.5 than there are ratings of 3 or 4, etc.).

```
#Rating movies
rating4 <- table(edx$rating)["4"]
rating35 <- table(edx$rating)["3.5"]
rating3 <- table(edx$rating)["3"]

Result <- (rating35 < rating3 && rating35 < rating4)

print(Result)

rm(rating35, rating3, rating4, Result)
```

```
[1] TRUE
```

Graphic Rating movies

```
#Graphic Rating movies
edx %>%
    group_by(rating) %>%
    summarize(count = n()) %>%
    ggplot(aes(x = rating, y = count)) +
    geom_line()
```

Plot mean movie ratings given by users

```
# Plot mean movie ratings given by users
edx %>%
  group_by(userId) %>%
  filter(n() >= 100) %>%
  summarize(b_u = mean(rating)) %>%
  ggplot(aes(b_u)) +
  geom_histogram(bins = 30, color = "yellow") +
  xlab("Mean rating") +
  ylab("Number of users") +
  ggtitle("Ratings by users") +
  scale_x_discrete(limits = c(seq(0.5,5,0.5))) +
  theme_light()
```

```
Warning message:
"Continuous limits supplied to discrete scale.
Did you mean `limits = factor(...)` or `scale_*_continuous()`?"
```

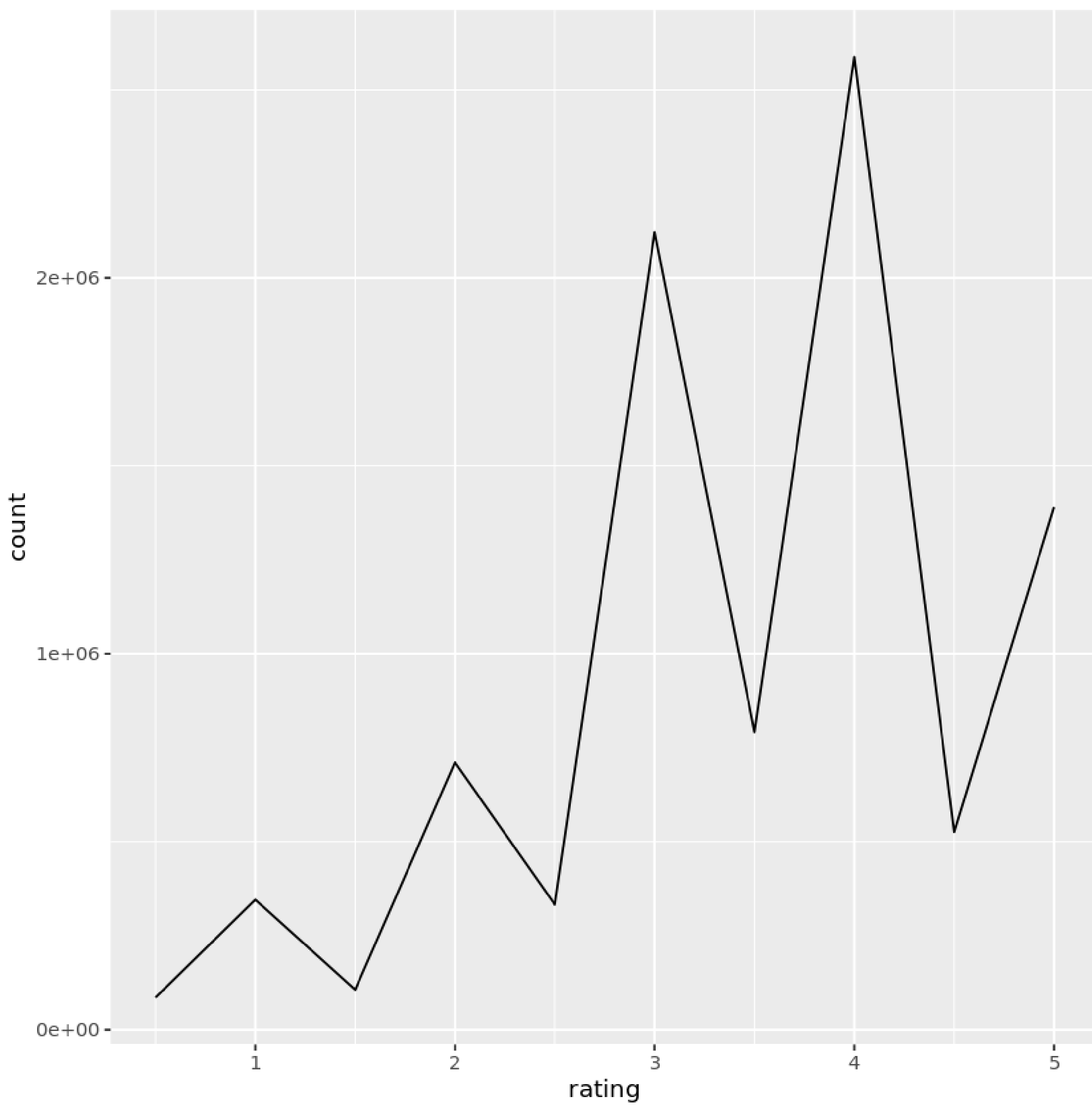


Figure 3: png

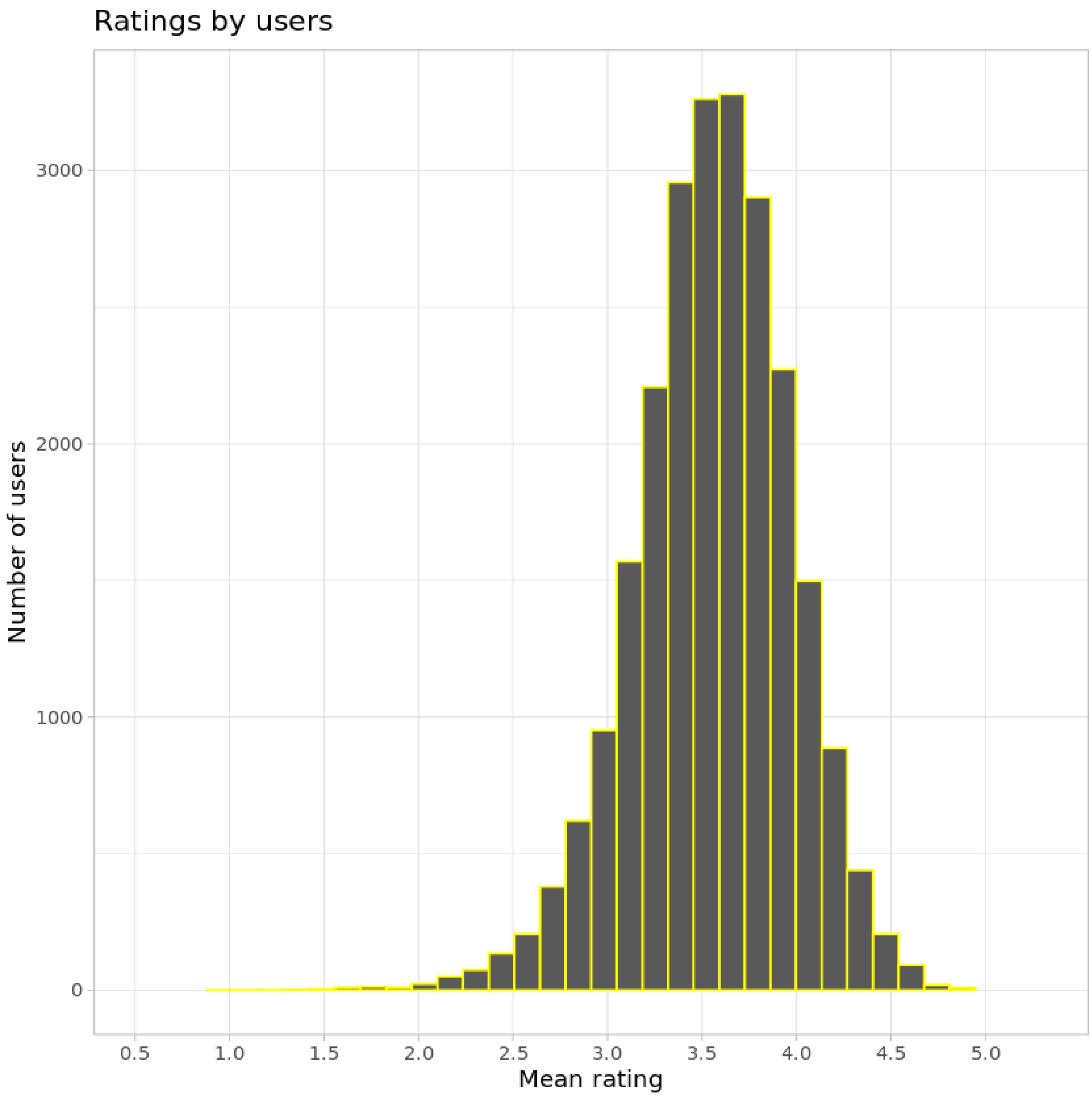

Ratings by users
Number of users
Mean rating
0
1000
2000
3000
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

Figure 4: png

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

```r
ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                      col.names = c("userId", "movieId", "rating", "timestamp"))
```

```r
#Create DAT for graphic and see categories by genres and years
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")
ratings <- read.table(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
                      col.names = c("userId", "movieId", "rating", "timestamp"))

dat <- movies %>% separate_rows(genres, sep ="\\|")
DAT.aggregate <- aggregate(formula = cbind(n = 1:nrow(dat)) ~ genres, data = dat, FUN = length)
```

```r
#Size of dataset
movielens <- left_join(ratings, movies, by = "movieId")
nrow(movielens)
```

10000054

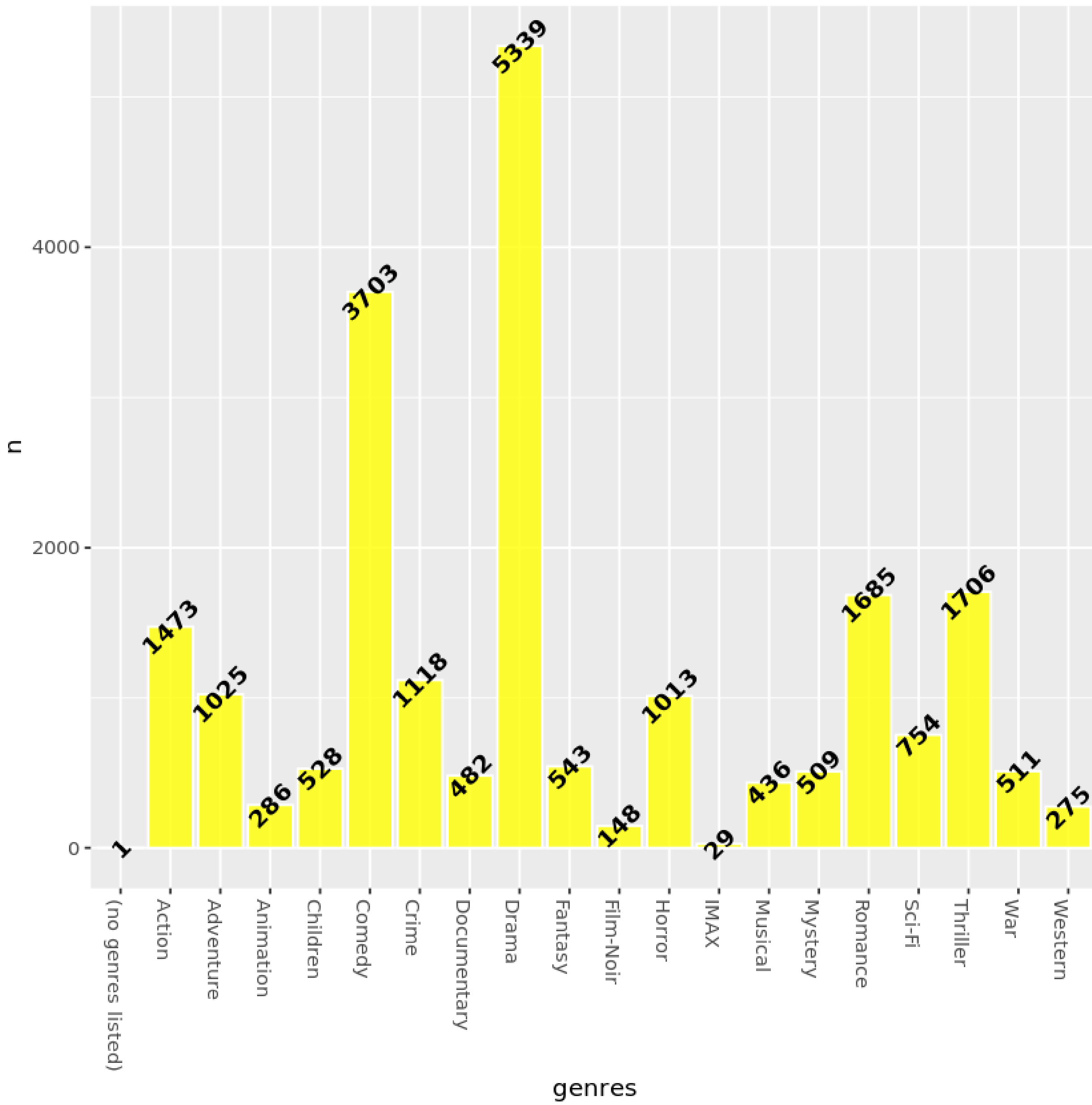
Genres as Drama and Comedy have high rating.

```r
#Genres as Drama and Comedy have high rating.

ggplot(DAT.aggregate,aes(x=genres,y=n))+
    geom_col(group="genres",alpha=0.8,fill ="yellow", color="white")+
    geom_text(label=DAT.aggregate$n,angle=45,fontface = "bold")+
theme(axis.text.x=element_text(angle = -90, hjust = 0))
```

```r
#Movies from the 1980s to 1990s and older have higher average ratings than recent movies.

movielens$year <- as.numeric(substr(as.character(movielens$title),
                  nchar(as.character(movielens$title))-4,
                  nchar(as.character(movielens$title))-1))
plot(table(movielens$year),
    col = "yellow")
```



n
4000
2000
0
1
1473
1025
286
528
3703
1118
482
5339
543
148
1013
29
436
509
1685
754
1706
511
275
(no genres listed)
Action
Adventure
Animation
Children
Comedy
Crime
Documentary
Drama
Fantasy
Film-Noir
Horror
IMAX
Musical
Mystery
Romance
Sci-Fi
Thriller
War
Western
genres

Figure 5: png

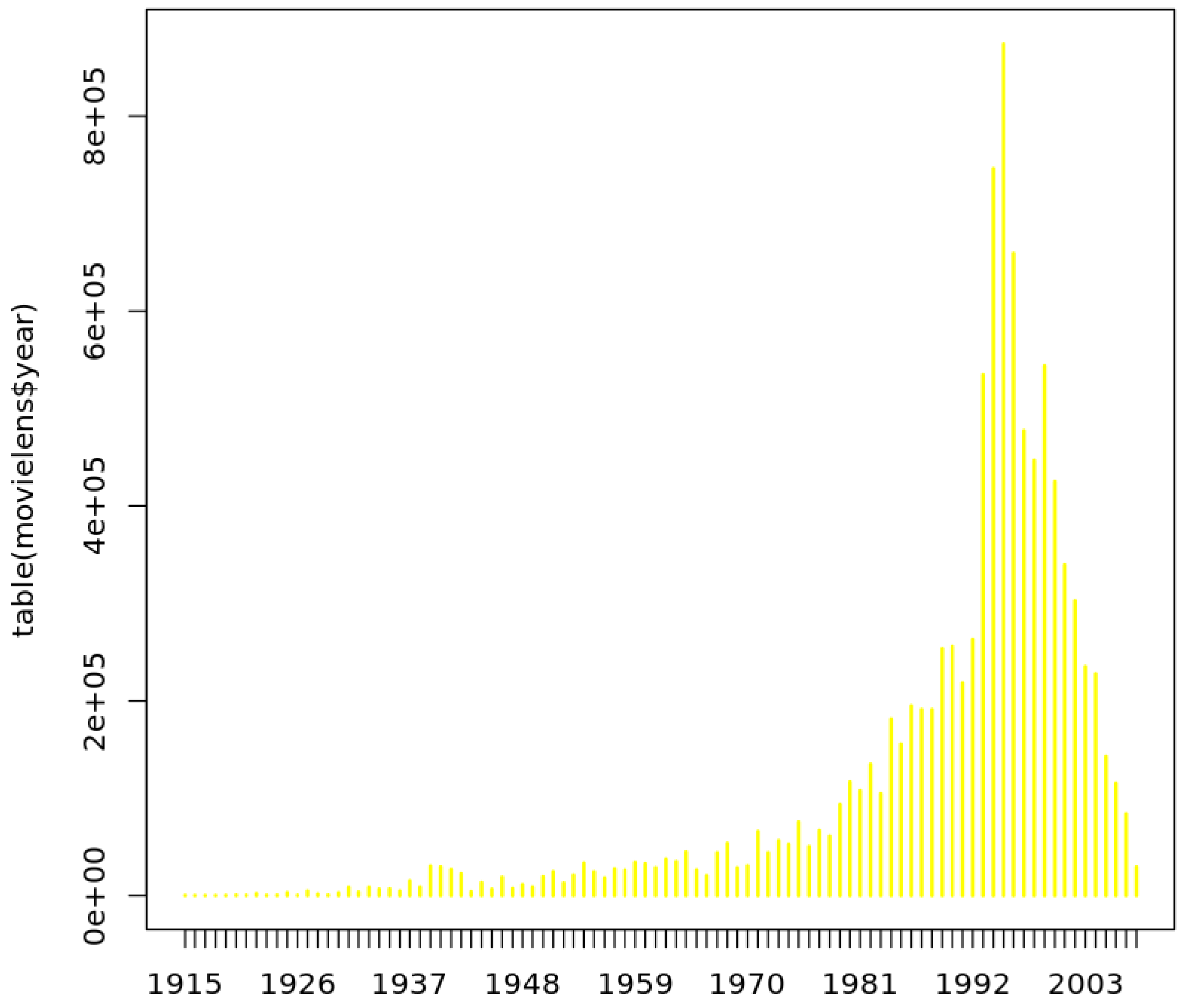

table(movielens$year)
0e+00
2e+05
4e+05
6e+05
8e+05
1915
1926
1937
1948
1959
1970
1981
1992
2003

Figure 6: png

4. MODELING

Predicted movie ratings and calculates RMSE.

Movie rating predictions will be compared to the true ratings in the validation set using RMSE

Model Approach

Movilens is a very large database with different variables that have different effects on ratings. Genres have a significant effect on ratings, it is required to divide compound genres into individual genres and calculate the effect of each genre using relatively more complex calculations. Some movies have a very high number of ratings, while others have very few or low ratings. Coming from small samples -few numbers of grades- can adversely affect preaching. we will use a method known as Regularization to penalize of very high or low grades that come from small samples. Also, we will divide the edx dataset into two parts: train (80%) and test (20%), then we will use train to train the model and test to cross-validate and fit the model to get the best lambda value that results in a minimum RMSE.

Created partition the data set Edx into 20% for test and 80% for training set, this step prepared the data split to create the model.

```
#Validation set will be 20% of the movieLens data
set.seed(2) #Partition dataset test and train
test_index <- createDataPartition(y = edx$rating, times = 2, p = 0.1, list = FALSE)
train <- edx[-test_index,] #train set
test <- edx[test_index,]   #test set
```

```
#Define RMSE that mesure of how spread out thesse residuals are, or concentration the data around the l

RMSE <- function(true_ratings, predicted_ratings){
sqrt(mean((true_ratings - predicted_ratings)^2,na.rm =TRUE))
}
```

```
mu <- mean(train$rating)  #mean

rmse_naive <- RMSE(test$rating, mu)  #MODEL 1

rmse_results <- data_frame(method='Average NAIVE', RMSE=rmse_naive)
rmse_results
```

method

RMSE

Average NAIVE

1.060381

```
#Model by Movie average

mu_m2 <-  mean(train$rating) #Model 2: Y u, i =?? + b i
movie_avgs <- train%>%
  group_by(movieId) %>%
```

```
  summarize(b_i=mean(rating-mu_m2))

predicted_rating <- mu_m2+test%>%
  left_join(movie_avgs, by='movieId')%>%
  pull(b_i)

rmse_m2 <- RMSE(predicted_rating, test$rating)
rmse_results <- bind_rows(rmse_results, data_frame(method='Movie Effect Model', RMSE=rmse_m2))
rmse_results
```

method

RMSE

Average NAIVE

1.0603807

Movie Effect Model

0.9435103

```
#Model by User average

user_avgs <- train%>%
  left_join(movie_avgs, by='movieId') %>% # Y u, i =??+b i +?? u, i w
  group_by(userId) %>%
  summarize(b_u=mean(rating - mu_m2 - b_i))

predicted_ratings <- test%>%
  left_join(movie_avgs, by='movieId')%>%
  left_join(user_avgs, by='userId')%>%
  mutate(pred=mu_m2 + b_i + b_u)%>%
  pull(pred)

rmse_m3 <- RMSE(predicted_ratings, test$rating)
rmse_results <- bind_rows(rmse_results, data_frame(method='Movie + User Effect Model', RMSE=rmse_m3))
rmse_results
```

method

RMSE

Average NAIVE

1.0603807

Movie Effect Model

0.9435103

Movie + User Effect Model

0.8660346

RMSE USED VALIDATION SET

```r
rating_vp <- validation %>%
  left_join(movie_avgs, by = 'movieId')%>%
  left_join(user_avgs, by='userId')%>%
  mutate(pred=mu_m2 + b_i + b_u)%>%
  pull(pred)

validation_m3 <- RMSE(validation$rating, rating_vp)
rmse_results <- bind_rows(rmse_results, data_frame(Method='Validation', RMSE=validation_m3))
rmse_results
```

method

RMSE

Method

Average NAIVE

1.0603807

NA

Movie Effect Model

0.9435103

NA

Movie + User Effect Model

0.8660346

NA

NA

NaN

Validation

```r
edx <- train %>% # It extracts the release year of the movie.
  mutate(title = str_trim(title)) %>%
  extract(title, c("title_tmp", "year"),
          regex = "^(.*) \\(([0-9 \\-]*)\\)$",
          remove = F) %>%
  mutate(year = if_else(str_length(year) > 4,
                        as.integer(str_split(year, "-",
                                             simplify = T)[1]),
                        as.integer(year))) %>%
  mutate(title = if_else(is.na(title_tmp), title, title_tmp)) %>%
  select(-title_tmp)  %>%
  mutate(genres = if_else(genres == "(no genres listed)",
                          `is.na<-`(genres), genres))
validation <- test %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")
```

```r
#Root Mean Square Error Loss Function
RMSE <- function(true_ratings, predicted_ratings){
        sqrt(mean((true_ratings - predicted_ratings)^2))
      }

lambdas <- seq(0, 5, 0.25)
rmses <- sapply(lambdas,function(l){

  #Mean of ratings from the edx training set
  mu <- mean(train$rating)

  #Adjust mean by movie effect and penalize low number on ratings
  b_i <- train %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  #Adjdust mean by user and movie effect and penalize low number of ratings
  b_u <- train %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))

  #Predict ratings in the training set to derive optimal penalty value 'lambda'
  predicted_ratings <-
    train %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred

  return(RMSE(predicted_ratings, train$rating))
})
plot(lambdas, rmses, #Graphic
    col = "blue")
```

```r
lambda <- 0.5

pred_y_lse <- sapply(lambda,function(l){

  #Derive the mearn from the training set
  mu <- mean(edx$rating)

  #Calculate movie effect with optimal lambda
b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

  #Calculate user effect with optimal lambda
b_u <- edx %>%
    left_join(b_i, by="movieId") %>%
    group_by(userId) %>%
    summarize(b_u = sum(rating - b_i - mu)/(n()+l))
```

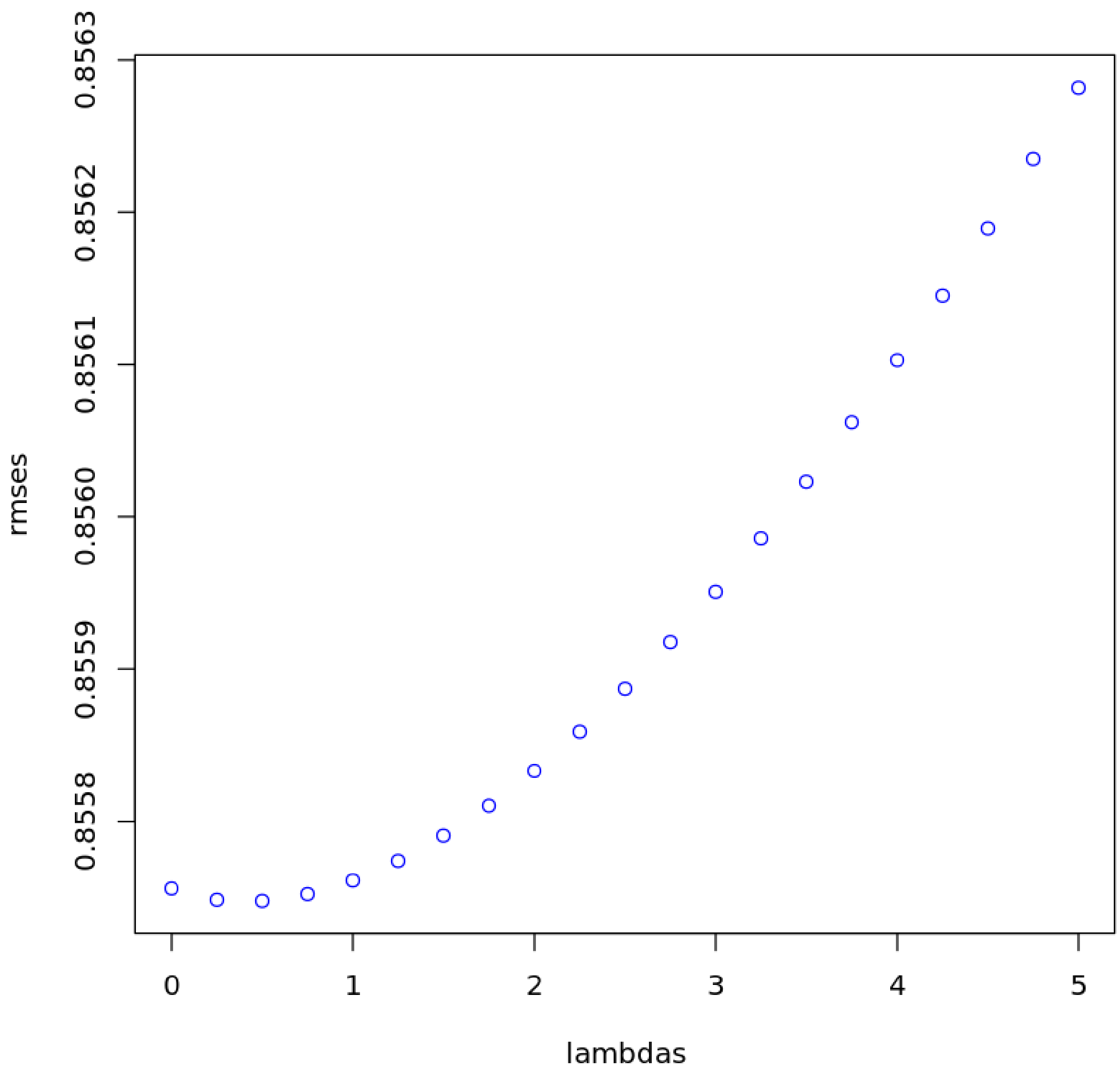


Figure 7: png

```
  #Predict ratings on validation set
predicted_ratings <-
    validation %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    mutate(pred = mu + b_i + b_u) %>%
    .$pred #validation

  return(predicted_ratings)

})
#Plot Linear, where we can see that the movies before 2000 are preferred for the customers.
avg_ratings <- edx %>%
group_by(year) %>%
summarise(avg_rating = mean(rating))
plot(avg_ratings)
```

```
library(lubridate)

tibble(`Initial Date` = date(as_datetime(min(edx$timestamp), origin="1980-01-01")),
       `Final Date` = date(as_datetime(max(edx$timestamp), origin="1980-01-01"))) %>%
  mutate(Period = duration(max(edx$timestamp)-min(edx$timestamp)))
```

```
Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

Initial Date

Final Date

Period

2005-01-08

2019-01-05

441479128s (~13.99 years)

```
edx %>% mutate(date = date(as_datetime(timestamp, origin="1990-01-01"))) %>%
  group_by(date, title) %>%
  summarise(count = n()) %>%
  arrange(-count) %>%
  head(25)
```

```
`summarise()` has grouped output by 'date'. You can override using the `.groups` argument.
```

date

title

count

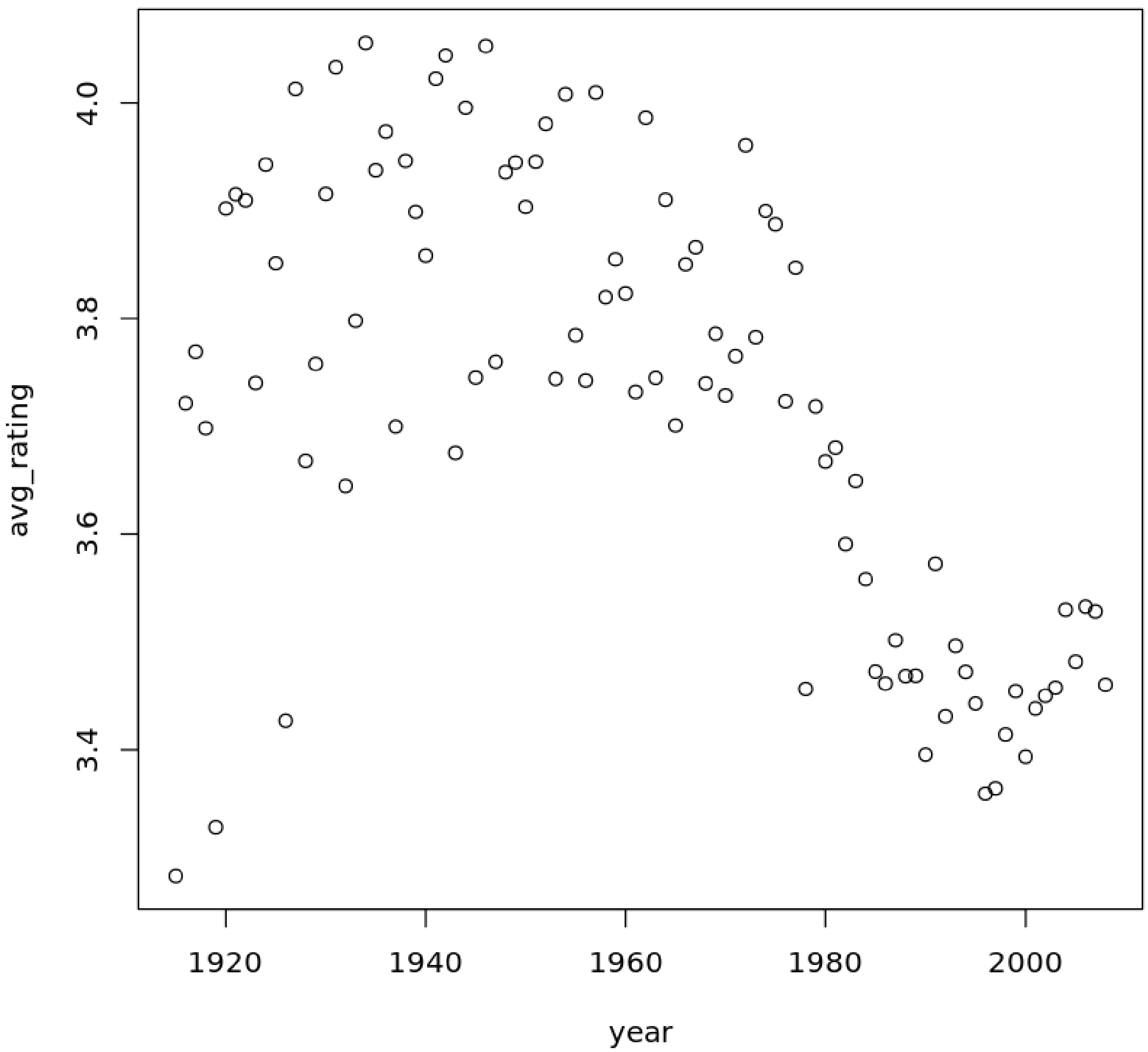


Figure 8: png

2018-05-22

Chasing Amy

266

2020-11-20

American Beauty

215

2019-12-11

Star Wars: Episode IV - A New Hope (a.k.a. Star Wars)

202

2019-12-11

Star Wars: Episode V - The Empire Strikes Back

200

2025-03-22

Lord of the Rings: The Two Towers, The

197

2019-12-11

Star Wars: Episode VI - Return of the Jedi

189

2025-03-22

Lord of the Rings: The Fellowship of the Ring, The

177

2020-11-20

Jurassic Park

173

2020-11-20

Terminator 2: Judgment Day

170

2019-12-11

Matrix, The

169

2020-11-20

Men in Black

167

2025-03-22

Shrek

158

2020-11-20

Star Wars: Episode IV - A New Hope (a.k.a. Star Wars)

157

2020-11-20

Star Wars: Episode V - The Empire Strikes Back

154

2020-11-20

Matrix, The

151

2019-12-11

Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark)

150

2020-11-20

Saving Private Ryan

150

2020-11-20

Star Wars: Episode VI - Return of the Jedi

150

2019-12-11

Saving Private Ryan

146

2019-12-11

E.T. the Extra-Terrestrial

145

2019-12-11

Godfather, The

145

2025-03-22

Fight Club

145

2025-03-22

Matrix, The

143

2020-11-20

Braveheart

142

2025-03-22

Shawshank Redemption, The

140

5 Data Analysis(EDA)

#

Features:

- Title of the movie.
- Year of realse and rated.
- Timestamp Information.

```
We will verify the division of the database, and thus compare with two graphs that will show us a bette
```

```
questions <- c("How many different movies are in the edx dataset?",
                "How many different genres are in the edx dataset?",
                "How many different titles are in the edx dataset?",
                "How many different users are in the edx dataset?",
                "What is the rating of movies per users?"
)
values_edx <- c(round(n_distinct(edx$movieId),0),
            round(n_distinct(edx$genres),0),
            round(n_distinct(edx$title),0),
            round(n_distinct(edx$userId),0),
            round(n_distinct(edx$movieId)*n_distinct(edx$userId)/dim(edx)[1] ,2)
)
values_validation <- c(round(n_distinct(validation$movieId),0),
                    round(n_distinct(validation$genres),0),
                    round(n_distinct(validation$title),0),
                    round(n_distinct(validation$userId),0),
                    round(n_distinct(validation$movieId)*n_distinct(validation$userId)/dim(edx)[1] ,2)
)
train_val <- data.frame(questions = questions, edx=values_edx, validation = values_validation )
train_val
```

questions

edx

validation

How many different movies are in the edx dataset?

10649.00

10169.00

How many different genres are in the edx dataset?

796.00

786.00

How many different titles are in the edx dataset?

10380.00

10168.00

How many different users are in the edx dataset?

69878.00

69692.00

What is the rating of movies per users?

102.07

97.21

```
edx <- edx %>% separate_rows(genres,sep = "\\|") %>% mutate(value=1)
        genres_rating <- edx %>% group_by(genres) %>% summarise(n=n())
        genres_rating <- genres_rating[order(-genres_rating$n),]
        ggplot(genres_rating, aes(x = n, y =reorder(genres, n),fill=genres)) +
        geom_bar(stat = "identity") +
        theme(legend.position = "none")+
        labs(title="Bar plot for Genres ratings", x="Count", y="Genres ratings")
```

We observe prevalence of genres, Drama continuing to be the most rated, and IMAX is the lowest, and give a correct cleaned the data deleted Nan, this model confirmed when before we agroup by genres.

```
edx %>% count(movieId) %>% ggplot(aes(n))+
        geom_histogram(color = "black" , fill= "green")+
        scale_x_log10()+
        labs( x = "log10 of count movieID")+
        ggtitle("Rating times Per Movie")+
        theme_gray()
```

```
‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```

```
install.packages("xtable", repos = "http://cran.us.r-project.org")
library(xtable)
```

```
Updating HTML index of packages in '.Library'
Making 'packages.html' ... done
```

Linear Regression Model Summary

```
model <- lm(rating ~ userId + movieId, data = edx)
result <- predict(model,validation)

lm_rmse <- RMSE(validation$rating, result)

model %>%
summary() %>%
xtable()
```

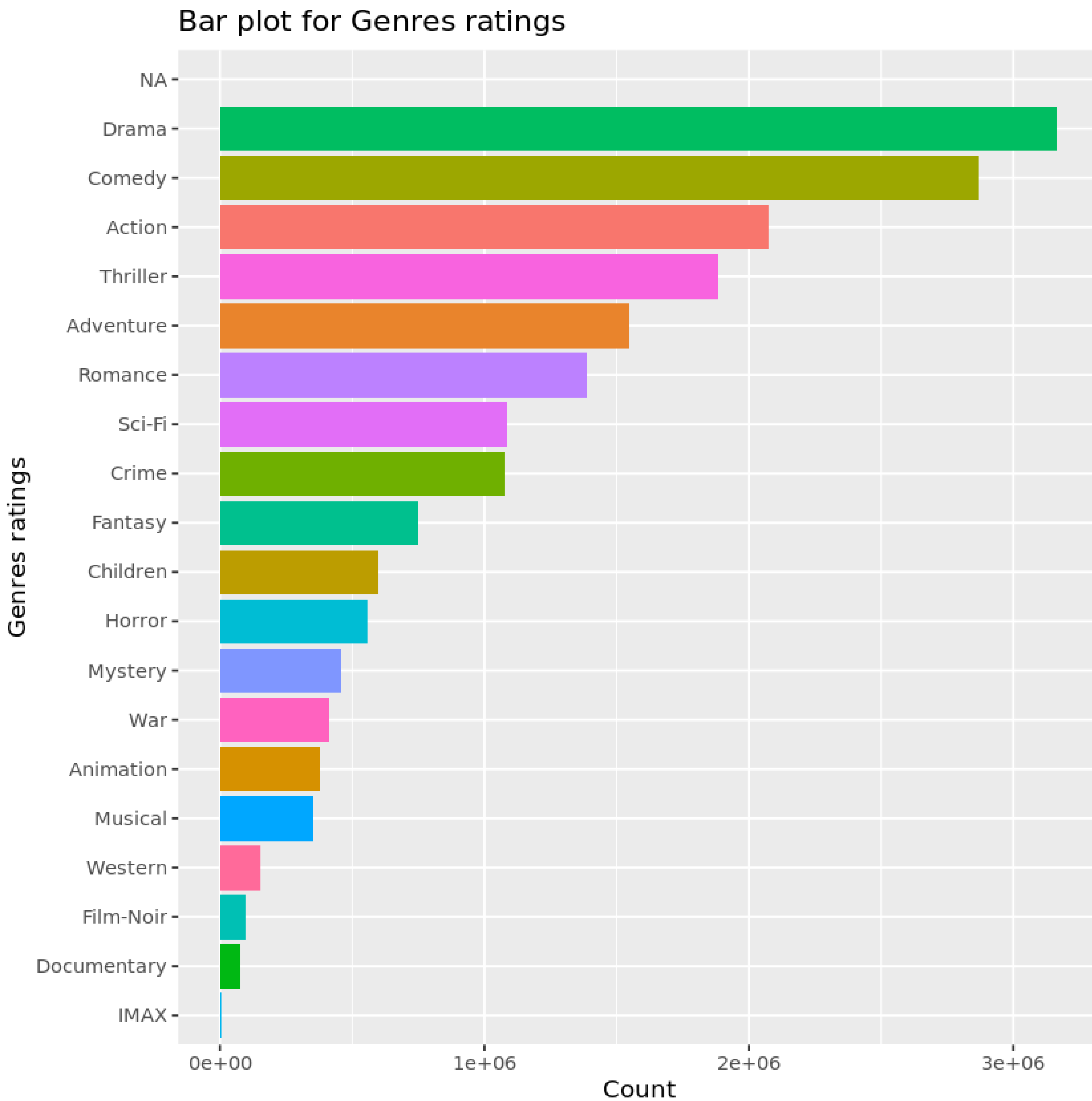

Bar plot for Genres ratings
NA
Drama
Comedy
Action
Thriller
Adventure
Romance
Sci-Fi
Crime
Fantasy
Children
Horror
Mystery
War
Animation
Musical
Western
Film-Noir
Documentary
IMAX
Genres ratings
0e+00
1e+06
2e+06
3e+06
Count

Figure 9: png

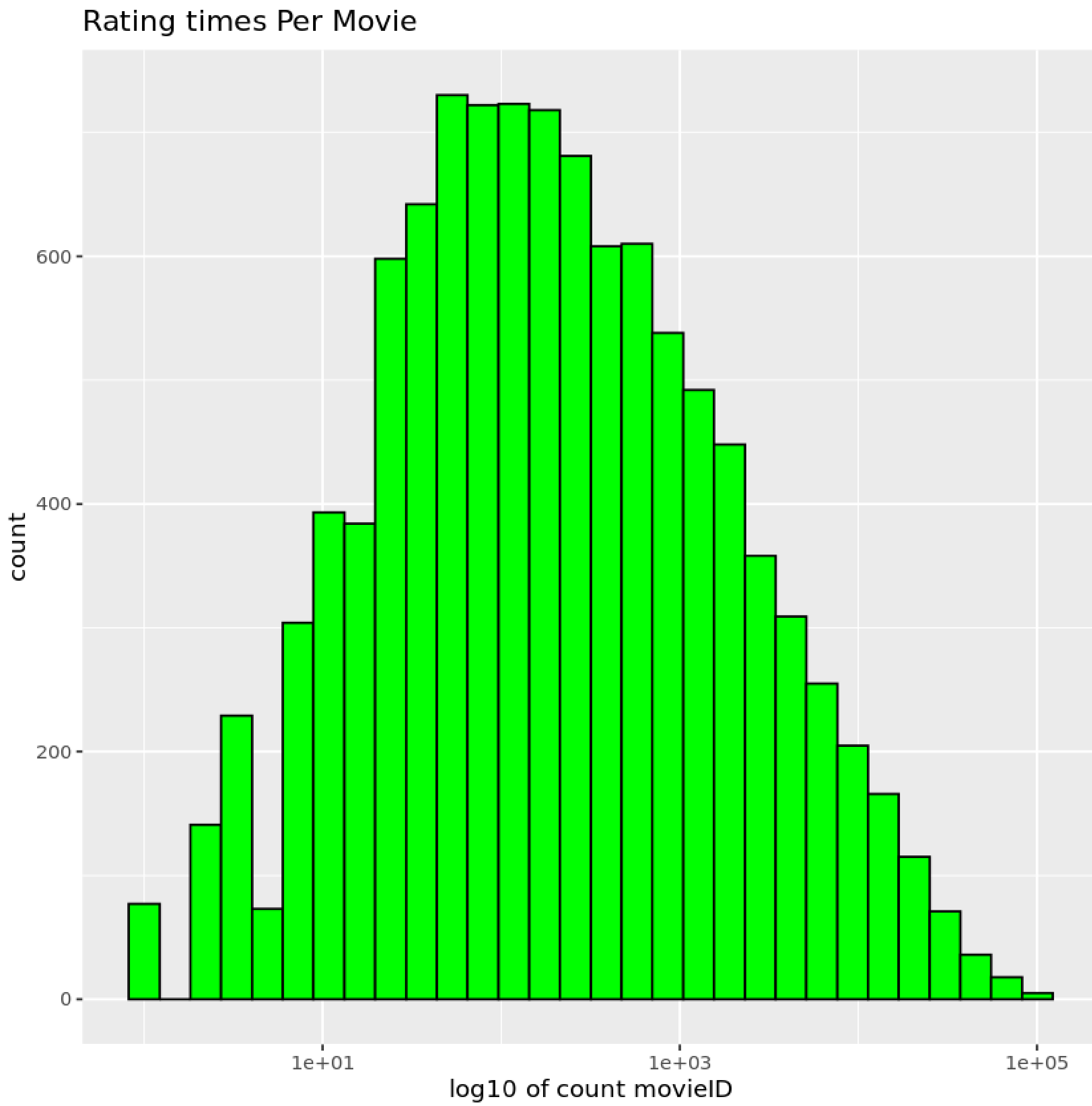


Figure 10: png

Estimate

Std. Error

t value

Pr(>|t|)

(Intercept)

3.525399e+00

4.981499e-04

7076.98374

0.000000e+00

userId

1.326330e-07

1.174836e-08

11.28950

1.479117e-29

movieId

-7.617538e-07

2.592977e-08

-29.37758

1.072640e-189

Histograms log10 of movieId looks like a gaussian distribution. From histograms, we observe the effects over Rating times by age at rate, userID and movieID, which are assimilated to some type of distribution, which allows us to approach an prediction by mean value.

The Model Linear Regression give us the same result of RMSE, with positive correlation that mean variables in which both variables move in the same direction.

```
library(Hmisc)
```

```
#Now we can see the correlationn in this 3 features,
edx_cor <- select(edx,c(rating,userId,movieId))
res_corr <- rcorr(as.matrix(edx_cor))
print(res_corr)
```

```
        rating userId movieId
rating    1.00      0   -0.01
userId    0.00      1    0.00
movieId  -0.01      0    1.00

n= 18931540


P
        rating userId movieId
rating         0      0
userId   0            0
movieId  0      0
```

CONCLUSION

With the different analyzes we can see how we discover the tastes of the spectators, as the films with the highest rating are not necessarily the most viewed, likewise it leads us to discover the taste for films of past decades over the most avant-garde ones.

RMSE, root-mean-square deviation, model that was used to predict from a list of rated movies, and discovers patterns. It was determined which movies viewers prefer, with a medium to high rating. (3 to 4). The films preferred by the clients were those produced from the periods 1920 and 2000 with a maximum rating of 4. The model yielded an accuracy of 86%, highlighting that the larger the sample size increases the accuracy or precision of the model and vice versa.