MovieLens - HarvardX: PH125.9x Data Science

Horea - Adrian Cioca

04/01/2021

MovieLens Project - A Rating Prediction For Movies

1. Introduction

The purpose of the MovieLens project is to create a recommendation system which will predict the user rating in order to be able to build a custom taste profile. We will start generating the data sets using the code provided by edx. The edx data set will be used for training our algorithm and the validation data set to predict movie ratings. RMSE(Root Mean Square Error) will be the indicator used to measure the error of the model in predicting the rating data. We used linear regression to predict the value of an outcome variable (rating) in base of one or more inputs predictors. To understand the behavior of the variables used in the linear model we used - Scatter plots to show the liner relationship between the predictor and response - Box plots to show any outlier observation in the variable - Density plots to show the distribution of the predictor variable like age of the movie, year of production, user id etc.

2. Data Setup

2.1 Create edx and validation set

Note: this process could take a couple of minutes If you don't have installed already the packages from if statements, they will be installed using specified repository (repos).

2.1.1 Data Download

2.1.1.0 Load Packages

```
## Loading required package: tidyr
```

```
## Loading required package: caret

## Loading required package: lattice

## Loading required package: ggplot2

## Loading required package: data.table

## Loading required package: dplyr

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

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

## Loading required package: broom

## Loading required package: 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:base':
##
##     date, intersect, setdiff, union

## Loading required package: sqldf

## Loading required package: gsubfn

## Loading required package: proto

## Loading required package: RSQLite

## Loading required package: e1071

## Loading required package: stringr

## Loading required package: stringi
```

2.1.1.1 Download MovieLens data

```
# 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"))
```

2.1.2 Create the Data Set

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

# if using R 3.6 or earlier:
movies <- as.data.frame(movies) %>% mutate(movieId =
as.numeric(levels(movieId))[movieId],
                                            title =
as.character(title),
                                            genres =
as.character(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")
```

Validation set will be 10% of MovieLens data

```
set.seed(1)
#set.seed(1, sample.kind="Rounding")
# 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)
# create the train set
edx <- movielens[-test_index,]
# create the test set
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)
```

Validation set will be used just to test the final algorithm

3. Analysis

3.1 General data analysis :

```
## Show the structure of edx
str(edx)

## Classes 'data.table' and 'data.frame':   9000061 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 838984474 838983653 838984885 838983707 ...
##  $ title    : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb &
Dumber (1994)" "Outbreak (1995)" ...
##  $ genres   : chr  "Comedy|Romance" "Action|Crime|Thriller"
"Comedy" "Action|Drama|Sci-Fi|Thriller" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

We have to check the integrity of the data if we have NAs we have to remove them So we do check each column

```
indx <- apply(edx, 2, function(x) any(is.na(x)))
indx

##    userId   movieId    rating timestamp     title    genres
##     FALSE     FALSE     FALSE     FALSE     FALSE     FALSE
```

We can see that our data is correct without NAs The head of the data:

```r
head(edx)
```

```
##    userId movieId rating timestamp                         title
## 1:      1     122      5 838985046              Boomerang (1992)
## 2:      1     185      5 838983525               Net, The (1995)
## 3:      1     231      5 838983392          Dumb & Dumber (1994)
## 4:      1     292      5 838983421               Outbreak (1995)
## 5:      1     316      5 838983392               Stargate (1994)
## 6:      1     329      5 838983392 Star Trek: Generations (1994)
##                           genres
## 1:                Comedy|Romance
## 2:         Action|Crime|Thriller
## 3:                        Comedy
## 4:  Action|Drama|Sci-Fi|Thriller
## 5:       Action|Adventure|Sci-Fi
## 6: Action|Adventure|Drama|Sci-Fi
```

3.1.1 The movie with the highest number of ratings

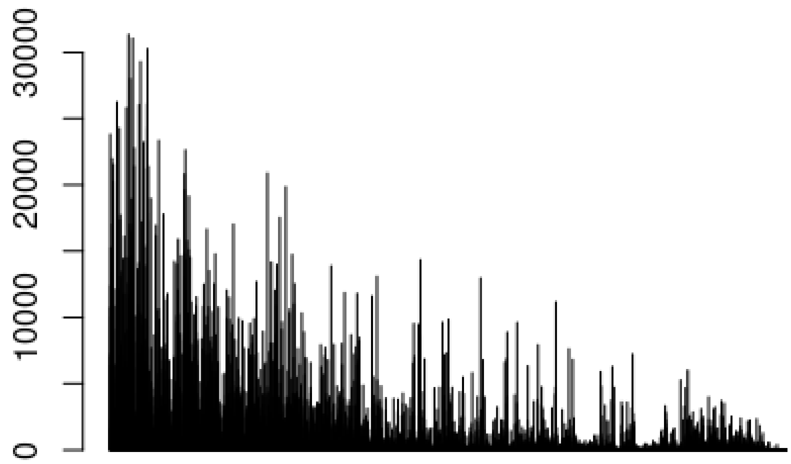
```r
## Return the movies with highest number of ratings
MovieIDRatings <-sqldf("select count(rating) as NumbersOfRatings,
movieId, title from edx group by movieId, title ")
head(MovieIDRatings)
```

```
##   NumbersOfRatings movieId                              title
## 1            23826       1                   Toy Story (1995)
## 2            10717       2                     Jumanji (1995)
## 3             7053       3            Grumpier Old Men (1995)
## 4             1579       4           Waiting to Exhale (1995)
## 5             6415       5 Father of the Bride Part II (1995)
## 6            12385       6                        Heat (1995)
```

So Toy Story has the higest number of ratings

3.1.2 A barplot of the ratings per movieId

```r
## Barplot of rtatings per movieId
barplot(MovieIDRatings$NumbersOfRatings)
```



3.1.3 The average ratings per movieId

```
## Calculate the average ratings per movieId
 avgMovieID <- mean(MovieIDRatings$NumbersOfRatings)
 avgMovieID

## [1] 842.9391
```

3.1.4 Genres of Movies

We want to count the ratings in functions of genres to be able to give a better prediction of the ratings. So first we retrieve the distinct genres from the database

```
## Extract the distinct genres from edx database
genres <- edx$genres %>% str_split(., pattern = "\\|")
genres <- genres %>% unlist() %>% unique()
genres

##  [1] "Comedy"             "Romance"            "Action"
##  [4] "Crime"              "Thriller"           "Drama"
##  [7] "Sci-Fi"             "Adventure"          "Children"
## [10] "Fantasy"            "War"                "Animation"
## [13] "Musical"            "Western"            "Mystery"
## [16] "Film-Noir"          "Horror"             "Documentary"
## [19] "IMAX"               "(no genres listed)"
```

Then the movie ratings per genres

We use sqldf library

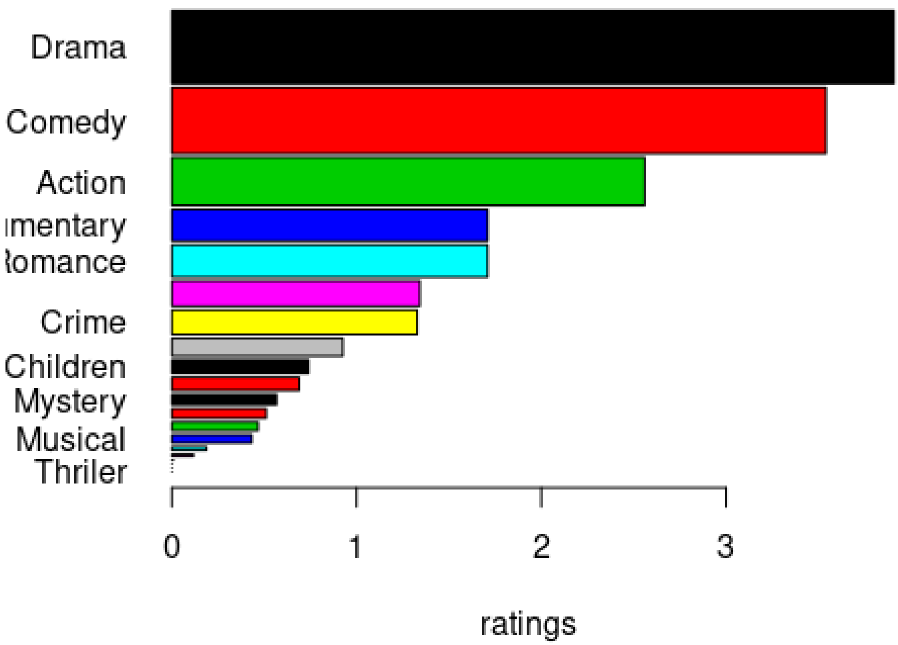
```
## Calculate the ratings per genres
Ratings_per_genres <- sqldf("select count(*) as rating, 'Comedy'
as genres  from edx where genres like '%Comedy%'  union
                             select count(*) as rating, 'Romance'
as genres  from edx where genres like '%Romance%' union
                             select count(*) as rating, 'Action'
as genres  from edx where genres like '%Action%' union
                             select count(*) as rating, 'Crime'
as genres  from edx where genres like '%Crime%' union
                             select count(*) as rating, 'Thriler'
as genres  from edx where genres like '%Thriler%' union
                             select count(*) as rating, 'Drama'
as genres  from edx where genres like '%Drama%' union
                             select count(*) as rating, 'SCi-Fi'
as genres  from edx where genres like '%Sci-Fi%' union
                             select count(*) as rating, 'Adventure'
as genres  from edx where genres like '%Adeventure%' union
                             select count(*) as rating, 'Children'
as genres  from edx where genres like '%Children%' union
                             select count(*) as rating, 'Fantasy'
as genres  from edx where genres like '%Fantasy%' union
                             select count(*) as rating, 'War'
as genres  from edx where genres like '%War%' union
                             select count(*) as rating, 'Animation'
as genres  from edx where genres like '%Animation%' union
                             select count(*) as rating, 'Musical'
as genres  from edx where genres like '%Musical%' union
                             select count(*) as rating, 'Western'
as genres  from edx where genres like '%Western%' union
                             select count(*) as rating, 'Mystery'
as genres  from edx where genres like '%Mystery%' union
                             select count(*) as rating, 'Film-Noir'
as genres  from edx where genres like '%Film-Noir%' union
                             select count(*) as rating, 'Horror'
as genres  from edx where genres like '%Horror%' union
                             select count(*) as rating, 'Documentary'
as genres  from edx where genres like '%Romance%' union
                             select count(*) as rating, 'IMAX'
as genres  from edx where genres like '%IMAX%' union
                             select count(*) as rating, 'NoGenres'
as genres  from edx where genres='' or genres is null")


# Save the data in descending order
Ratings_per_genres <- sqldf("select *  from Ratings_per_genres order
```

```
by rating desc" )
Ratings_per_genres
```

```
##     rating      genres
## 1  3909401       Drama
## 2  3541284      Comedy
## 3  2560649      Action
## 4  1712232 Documentary
## 5  1712232     Romance
## 6  1341750      SCi-Fi
## 7  1326917       Crime
## 8   925624     Fantasy
## 9   737851    Children
## 10  691407      Horror
## 11  567865     Mystery
## 12  511330         War
## 13  467220   Animation
## 14  432960     Musical
## 15  189234     Western
## 16  118394   Film-Noir
## 17    8190        IMAX
## 18       0   Adventure
## 19       0    NoGenres
## 20       0     Thriler
```

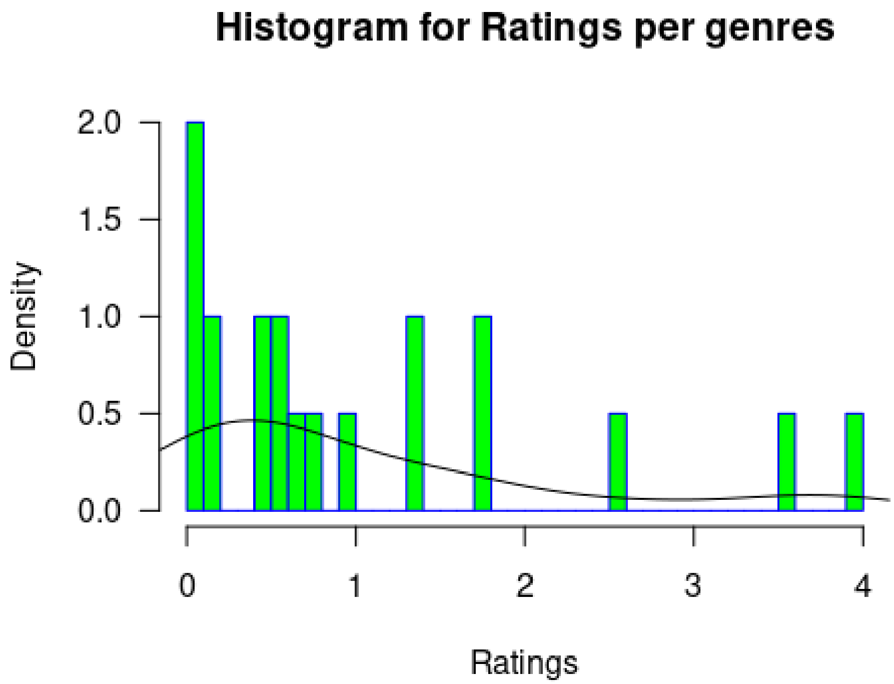
3.1.5 A barplot of genres in function of ratings:

```
## Barplot of genres in function of ratinngs
barplot(Ratings_per_genres$rating/1000000, names.arg =
Ratings_per_genres$genres,
 xlab = "ratings", horiz=TRUE, las=1, col=c(1:10),
 desc(Ratings_per_genres$rating/1000000))
```

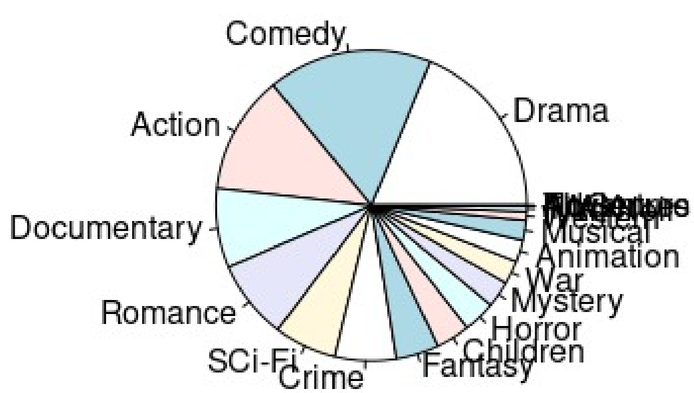


3.1.6 A histogram of ratings with the density:

```
## Histogram of ratings
hist(Ratings_per_genres$rating/1000000,
     main="Histogram for Ratings per genres",
     xlab="Ratings",
     border="blue",
     col="green",
     xlim=c(0,4),
     las=1,
     breaks=50,
     prob = TRUE)

lines(density(Ratings_per_genres$rating/1000000))
```



3.1.7 A pie of ratings counts per genres is bellow.



We can see from the pie chart the first three most rated genres : Drama, Comedy and Action We need also to count the distinct genres, movies and users:

```
##   distinct_genres distinct_movies distinct_users
## 1             797           10677          69878
```

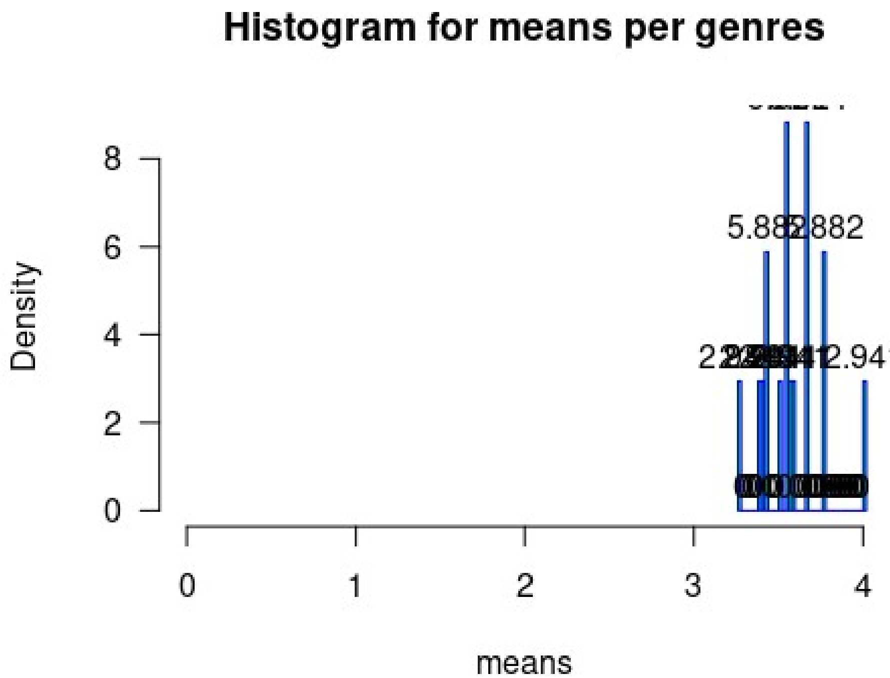
3.1.8 The mean for all genres

The mean for all genres and the mean for each genres will give us an overview of the movies in terms of the ratings:

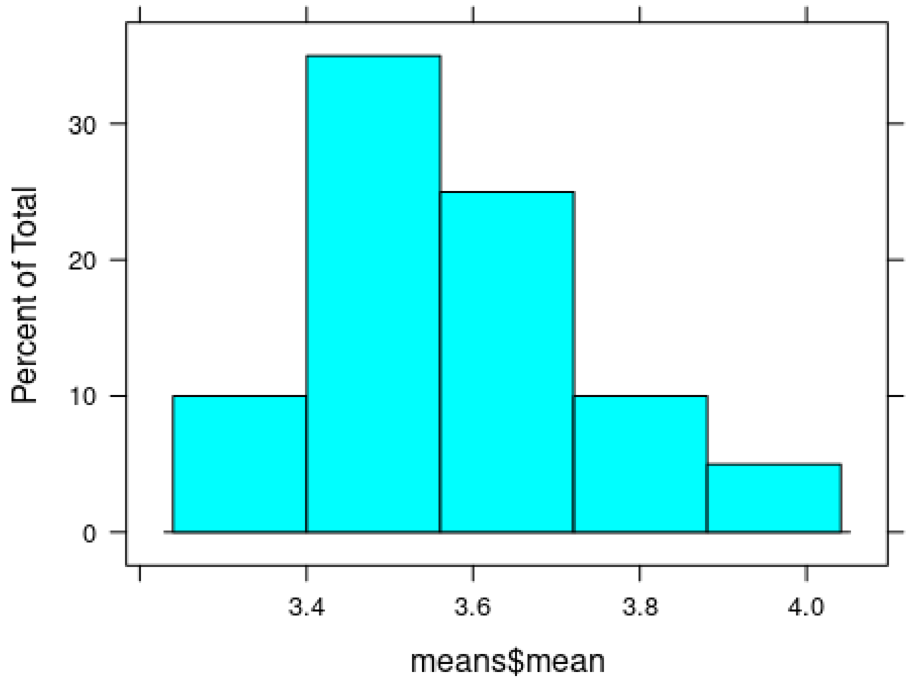
```
## [1] 3.512464
```

3.1.9 The mean per gender with histogram

```
##        mean      genres
## 1        NA   Adventure
## 2        NA    NoGenres
## 3        NA     Thriler
## 4  3.269523      Horror
## 5  3.396756      SCi-Fi
## 6  3.418673    Children
## 7  3.421589      Action
## 8  3.437040      Comedy
## 9  3.502419     Fantasy
## 10 3.553594 Documentary
## 11 3.553594     Romance
## 12 3.555122     Western
## 13 3.562761     Musical
## 14 3.599588   Animation
## 15 3.666151       Crime
## 16 3.673047       Drama
## 17 3.677412     Mystery
## 18 3.761844        IMAX
## 19 3.779457         War
## 20 4.011732   Film-Noir
```




```
## Warning in histogram.numeric(means$mean, means$genres): explicit
'data'
## specification ignored
```



3.1.10 Ratings function of date

Also we want to know the number of ratings in function of the date So we add a column date in the format yyyy-mm-dd which is a conversion of the timestamp column The new structure will be as follow:

```
## Classes 'data.table' and 'data.frame':   9000061 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 ...
##  $ date   : Date, format: "1996-08-02" "1996-08-02" ...
##  $ title  : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb &
Dumber (1994)" "Outbreak (1995)" ...
##  $ genres : chr  "Comedy|Romance" "Action|Crime|Thriller" "Comedy"
"Action|Drama|Sci-Fi|Thriller" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Now we can extract the released year from the title and saved as releasedYear :

```
## Classes 'data.table' and 'data.frame':   9000061 obs. of  7
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 ...
##  $ date         : Date, format: "1996-08-02" "1996-08-02" ...
##  $ title        : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb &
```

```
Dumber (1994)" "Outbreak (1995)" ...
##  $ genres      : chr  "Comedy|Romance" "Action|Crime|Thriller"
"Comedy" "Action|Drama|Sci-Fi|Thriller" ...
##  $ releasedYear: chr  "1992" "1995" "1994" "1995" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

Also the age of the movies will be calculated in base of the ReleasedYear:

```
## Classes 'data.table' and 'data.frame':   9000061 obs. of  8
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 ...
##  $ date        : Date, format: "1996-08-02" "1996-08-02" ...
##  $ title       : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb &
Dumber (1994)" "Outbreak (1995)" ...
##  $ genres      : chr  "Comedy|Romance" "Action|Crime|Thriller"
"Comedy" "Action|Drama|Sci-Fi|Thriller" ...
##  $ releasedYear: chr  "1992" "1995" "1994" "1995" ...
##  $ age         : num  29 26 27 26 27 27 27 27 27 27 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

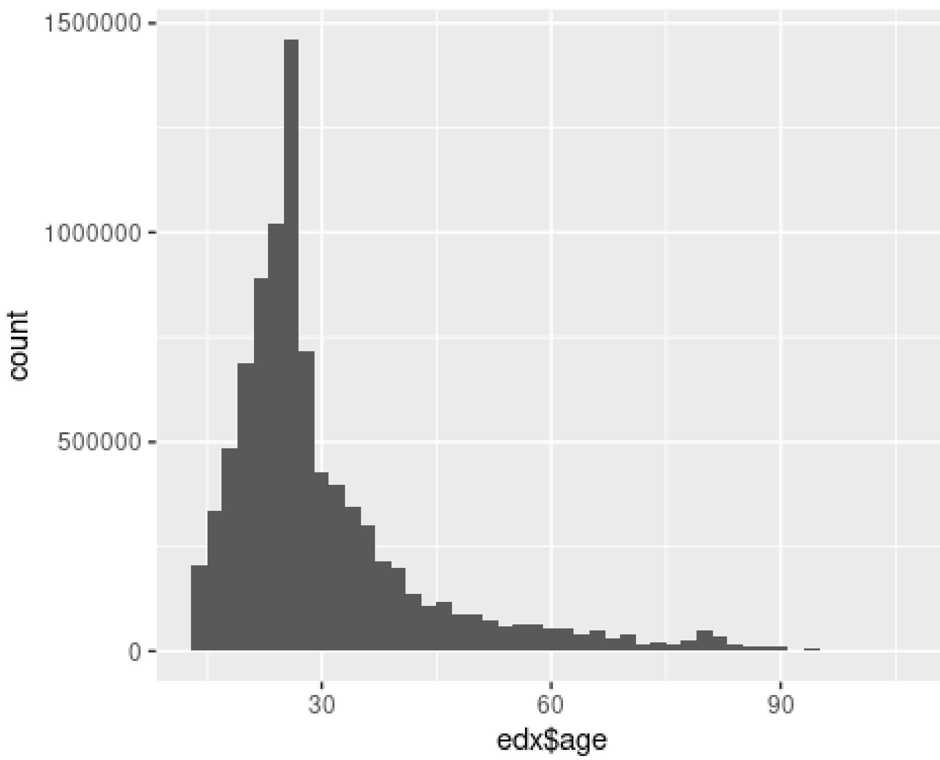
It’s important to have in our analyses the rating year This will be obtain as folow :

3.1.11 The minimum , maximum and the mean of the ages of the movies in edx:

```
## [1] 13

## [1] 106

## [1] 30.77898

## [1] "For Validation database:"

## [1] 13

## [1] 106

## [1] 30.79325
```

3.1.12 Histogram of ages

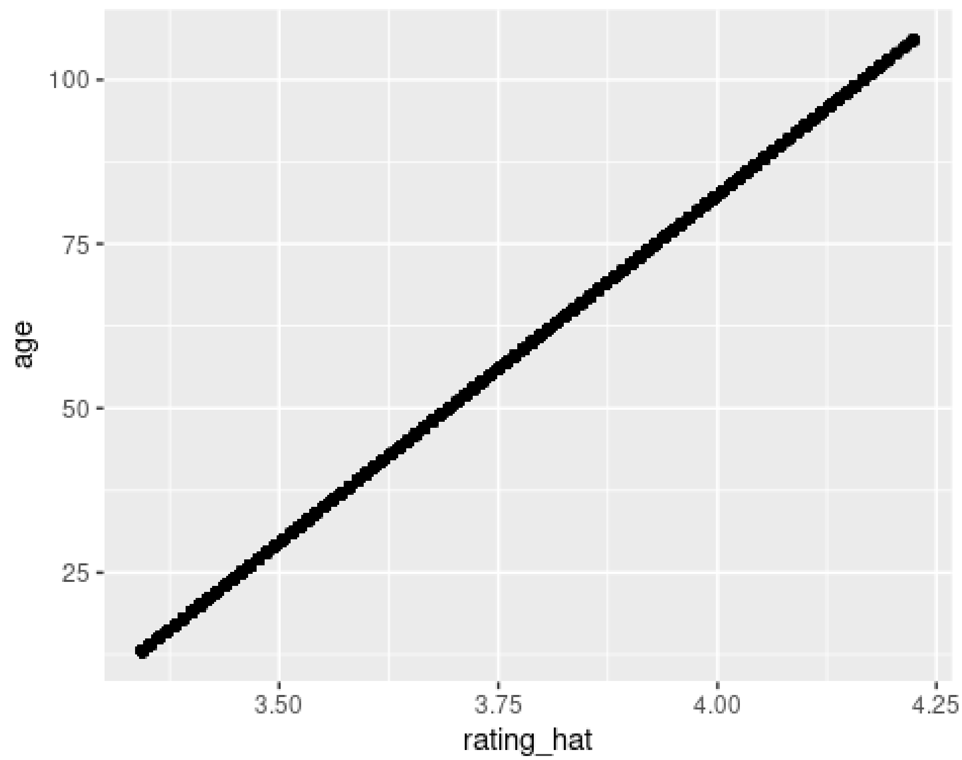
A histogram of ages show us that the highest number of movies from netflix have an age around 24 years:




```
## geom_bar: na.rm = FALSE, orientation = NA
## stat_bin: binwidth = NULL, bins = NULL, na.rm = FALSE, orientation
= NA, pad = FALSE
## position_stack
```

3.1.13 A plot of ratings and age on validation set

```
## # A tibble: 2 x 7
##   term        estimate std.error statistic p.value conf.low
conf.high
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
<dbl>
## 1 (Intercept)  3.22    0.00261       1236.       0  3.22      3.23

## 2 age          0.00946 0.0000774      122.       0  0.00931
0.00961
```



We will create also for test purpose a data base with the movies age under 30 years edx_30_minus

```
## Classes 'data.table' and 'data.frame':   5803301 obs. of  9 variables:
##  $ userId      : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId     : num  122 185 231 292 316 329 355 356 362 364 ...
##  $ rating      : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ date        : Date, format: "1996-08-02" "1996-08-02" ...
##  $ title       : chr  "Boomerang (1992)" "Net, The (1995)" "Dumb & Dumber (1994)" "Outbreak (1995)" ...
##  $ genres      : chr  "Comedy|Romance" "Action|Crime|Thriller" "Comedy" "Action|Drama|Sci-Fi|Thriller" ...
##  $ releasedYear: chr  "1992" "1995" "1994" "1995" ...
##  $ age         : num  29 26 27 26 27 27 27 27 27 27 ...
##  $ rating_year : num  1996 1996 1996 1996 1996 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

4. Linear Models Analysis

Like methods will be use : We will use edx database to train the model and edx_30_minus and validation databases to test the model

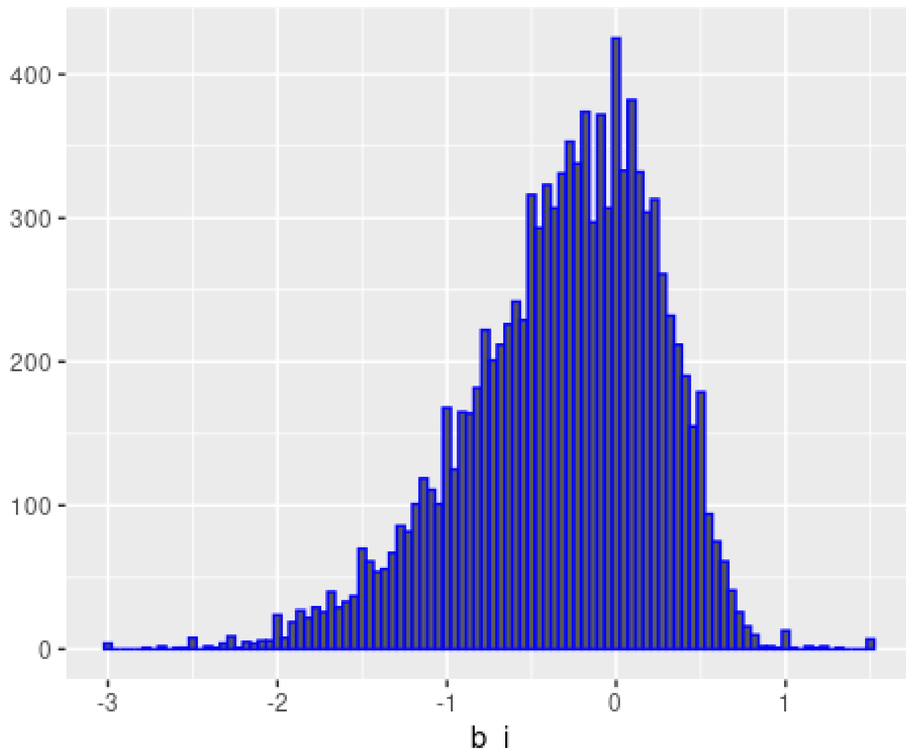
4.1 Mean effect model

```
## [1] 3.512464
```

```
## [1] 3.439311

## [1] 3.512044
```

4.2 Movie effect model

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Predicted ratings on validation database

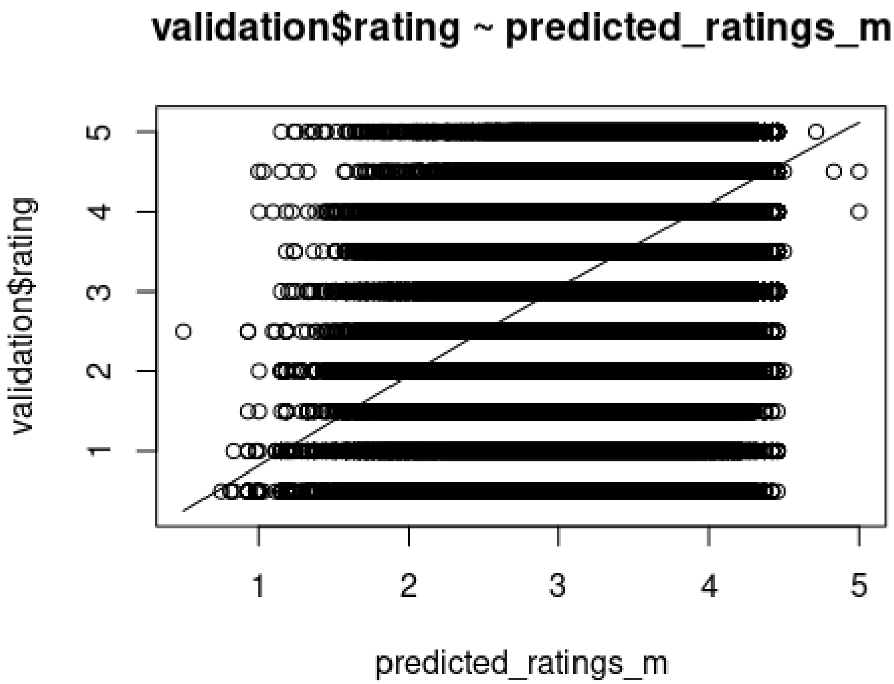
```
## [1] 3.669204 3.998851 2.943630 3.537779 4.072319 3.508718
```

Predicted ratings on edx_30_minus

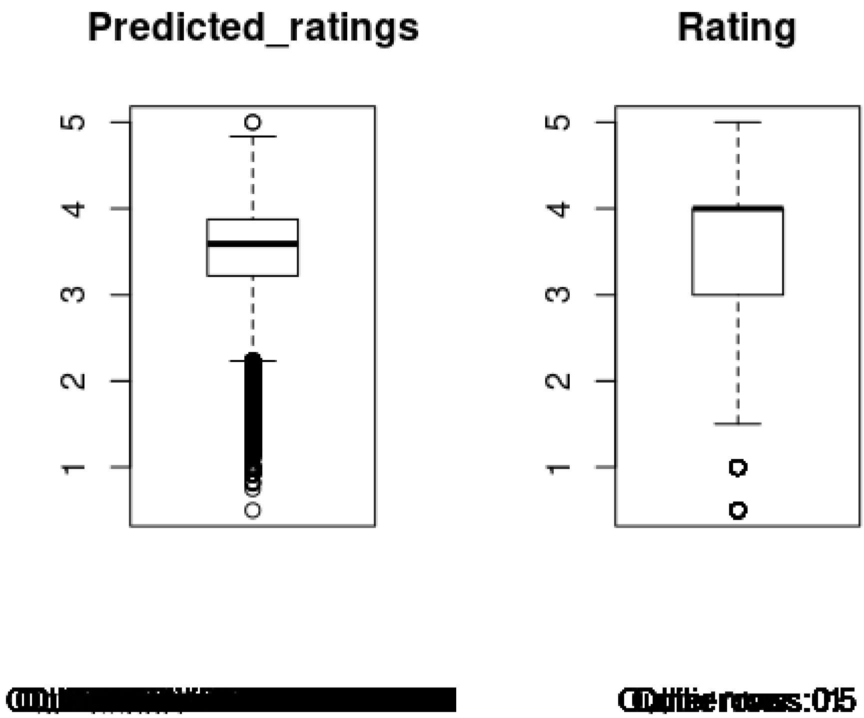
```
## [1] 2.863236 3.129984 2.936825 3.416580 3.350417 3.333678
```

Movie Analyse Model on validation database

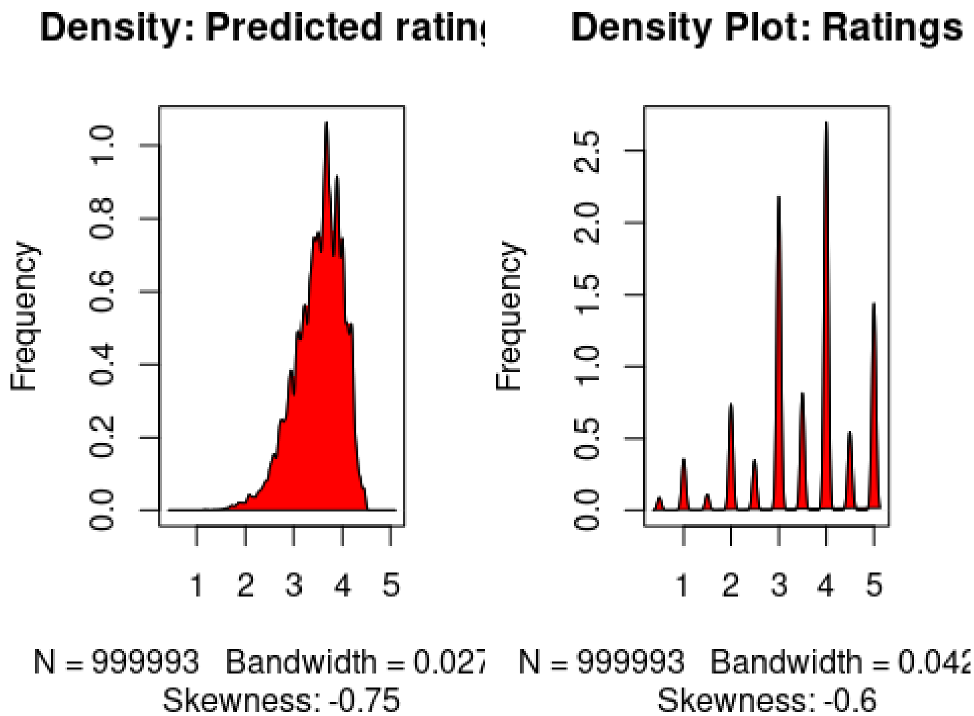
```
# Scatter plot
scatter.smooth(x=predicted_ratings_m, y=validation$rating,
main="validation$rating ~ predicted_ratings_m")
```




```
# Box plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
boxplot(predicted_ratings_m, main="Predicted_ratings",
sub=paste("Outlier rows: ", boxplot.stats(predicted_ratings_m)$out))
# box plot for 'predicted ratings'
boxplot(validation$rating, main="Rating", sub=paste("Outlier rows: ",
boxplot.stats(validation$rating)$out))  # box plot for 'validation
ratings'
```




```r
# Density plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
plot(density(predicted_ratings_m), main="Density: Predicted ratings", ylab="Frequency", sub=paste("Skewness:", round(e1071::skewness(predicted_ratings_m), 2)))  # density plot for 'Predicted ratings'
polygon(density(predicted_ratings_m), col="red")
plot(density(validation$rating), main="Density Plot: Ratings", ylab="Frequency", sub=paste("Skewness:", round(e1071::skewness(validation$rating), 2)))  # density plot for 'Ratings'
polygon(density(validation$rating), col="red")
```

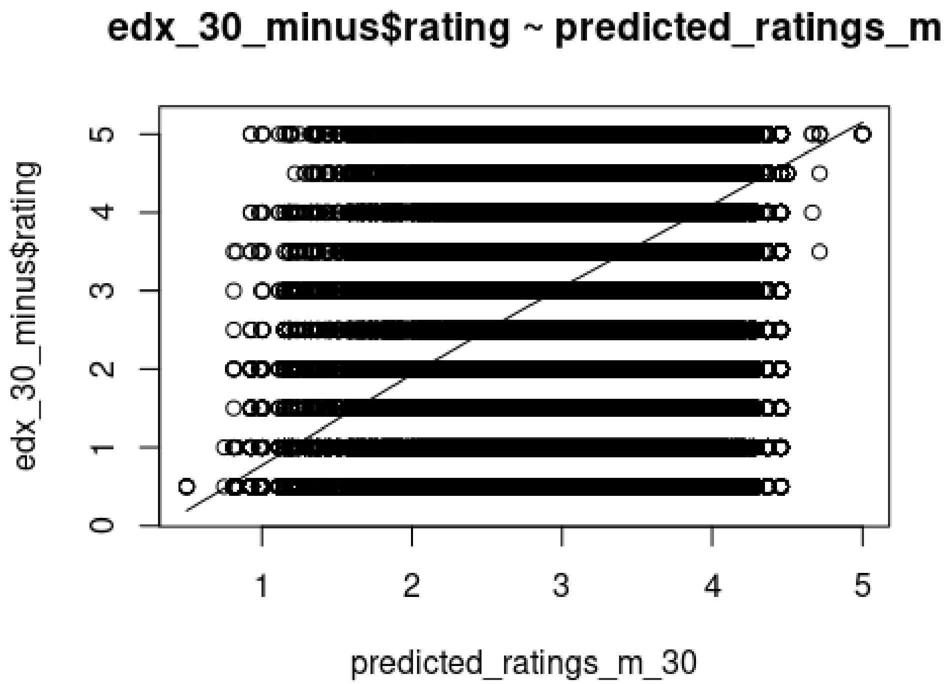



```r
# Correlation coefficient
print(cor(predicted_ratings_m, validation$rating))

## [1] 0.4564716
```

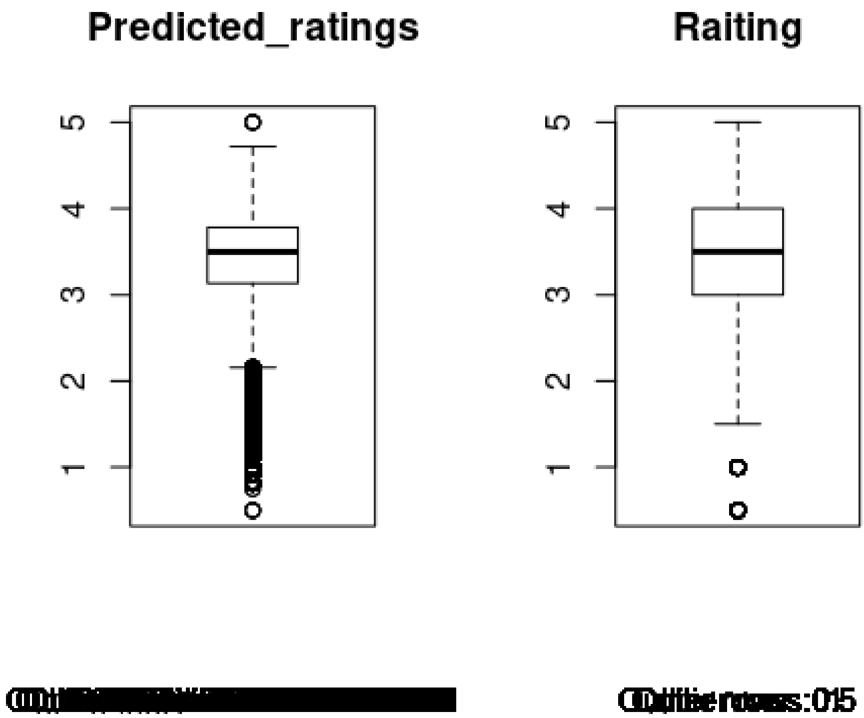
Density plot in general is showing how close we are to normality. Correlation show the linear dependence between two variables. It is between -1 and 1 If is positive closed to 1 it shows a strong dependence between the two variables. If is closed to 0 it shows a week dependence. In this case is bigger than 0 but not so closed to 1 is under 0.5. So we can say that is still a week dependency between these two variables

Movie Analyse Model on edx_30_minus database

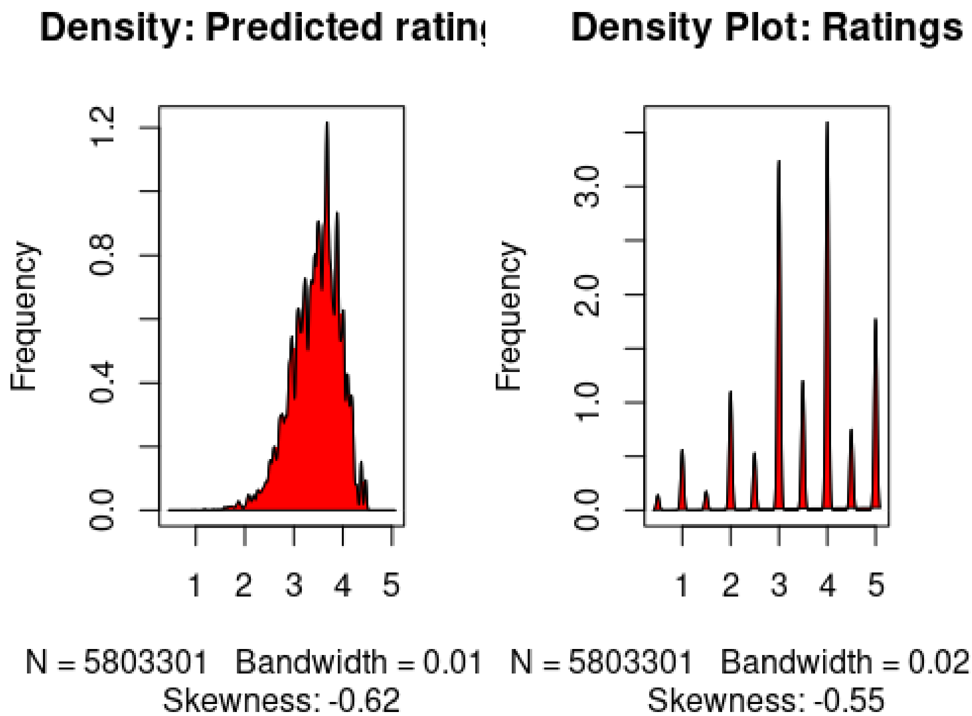
```r
# Scatter plot
scatter.smooth(x=predicted_ratings_m_30, y=edx_30_minus$rating,
main="edx_30_minus$rating ~ predicted_ratings_m")
```




```r
# Box plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
boxplot(predicted_ratings_m_30, main="Predicted_ratings",
sub=paste("Outlier rows: ",
boxplot.stats(predicted_ratings_m_30)$out))  # box plot for 'predicted
ratings'
boxplot(edx_30_minus$rating, main="Raiting", sub=paste("Outlier rows:
", boxplot.stats(edx_30_minus$rating)$out))  # box plot for
'validation ratings'
```




```r
# Density plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
plot(density(predicted_ratings_m_30), main="Density: Predicted
ratings", ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(predicted_ratings_m_30), 2)))  # density plot
for 'Predicted ratings'
polygon(density(predicted_ratings_m_30), col="red")
plot(density(edx_30_minus$rating), main="Density Plot: Ratings",
ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(edx_30_minus$rating), 2)))  # density plot for
'Ratings'
polygon(density(edx_30_minus$rating), col="red")
```




```
# Correlation coefficient
print(cor(predicted_ratings_m_30, edx_30_minus$rating))

## [1] 0.4459372
```

Build the linear model for Movies

On validation database

```
##
## Call:
## lm(formula = predicted_ratings_m ~ validation$rating, data =
validation)
##
## Coefficients:
##       (Intercept)  validation$rating
##            2.7763             0.2095
```

Now we can write the linear model formula for Movie Model
On validation database: predicted_ratings_m <- 2.7761 + 0.2096 *
validation$rating

On edx_30_minus

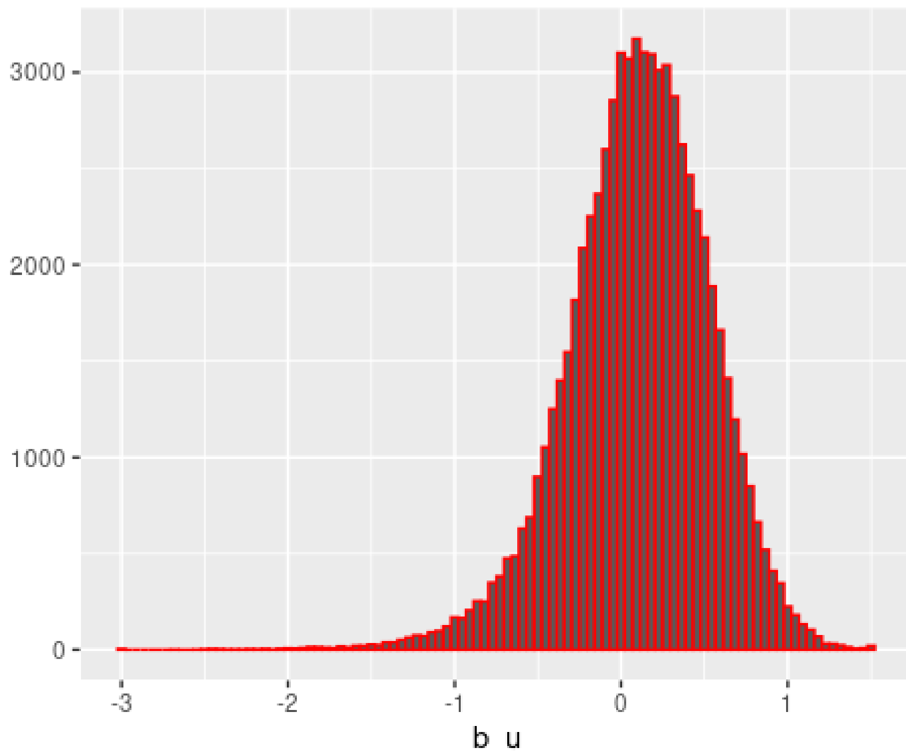
```
##
## Call:
## lm(formula = predicted_ratings_m_30 ~ edx_30_minus$rating, data =
edx_30_minus)
```

```
##
## Coefficients:
##         (Intercept)  edx_30_minus$rating
##              2.7554               0.1989
```

So the linear model formula for Movie Model on edx_30_minus
predicted_ratings_m <- 2.7558 + 0.1998 * validation$rating

4.3 User effect model

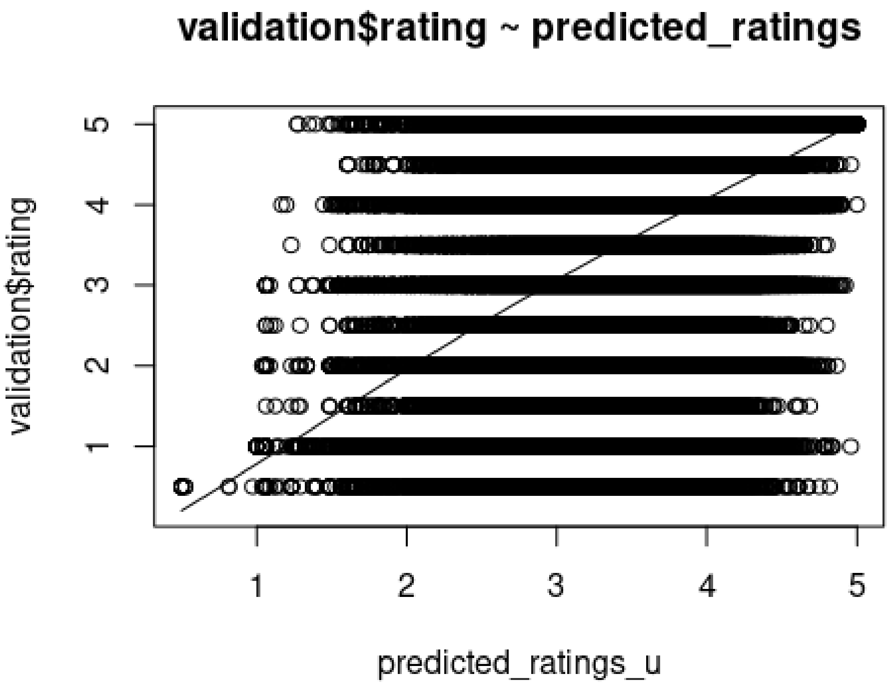
```
## `summarise()` ungrouping output (override with `.groups` argument)
```



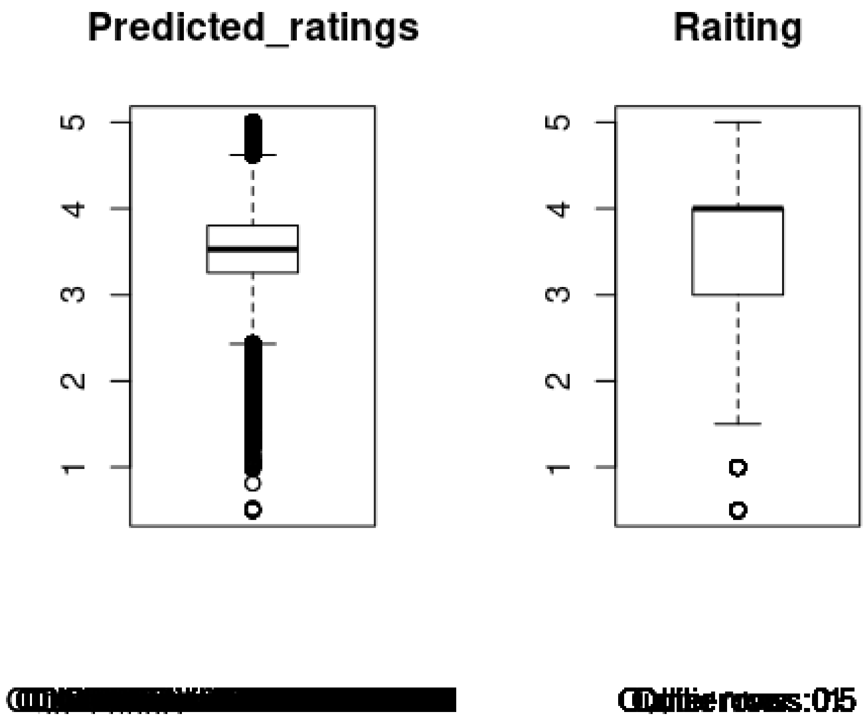
Predicted Ratings on validation database

```
## [1] 5.000000 3.166667 3.166667 3.983333 3.983333 3.983333
```

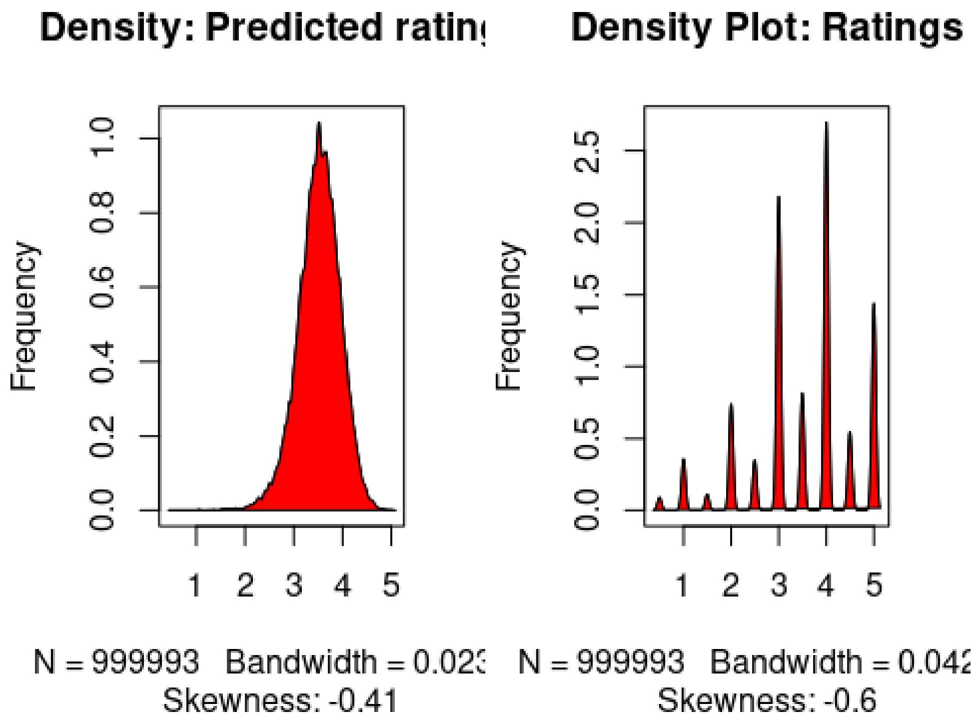
```
#Scatter plot
scatter.smooth(x=predicted_ratings_u, y=validation$rating,
main="validation$rating ~ predicted_ratings")
```




```
#Box plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
boxplot(predicted_ratings_u, main="Predicted_ratings",
sub=paste("Outlier rows: ", boxplot.stats(predicted_ratings_u)$out))
# box plot for 'predicted ratings'
boxplot(validation$rating, main="Raiting", sub=paste("Outlier rows: ",
boxplot.stats(validation$rating)$out))  # box plot for 'validation
ratings'
```



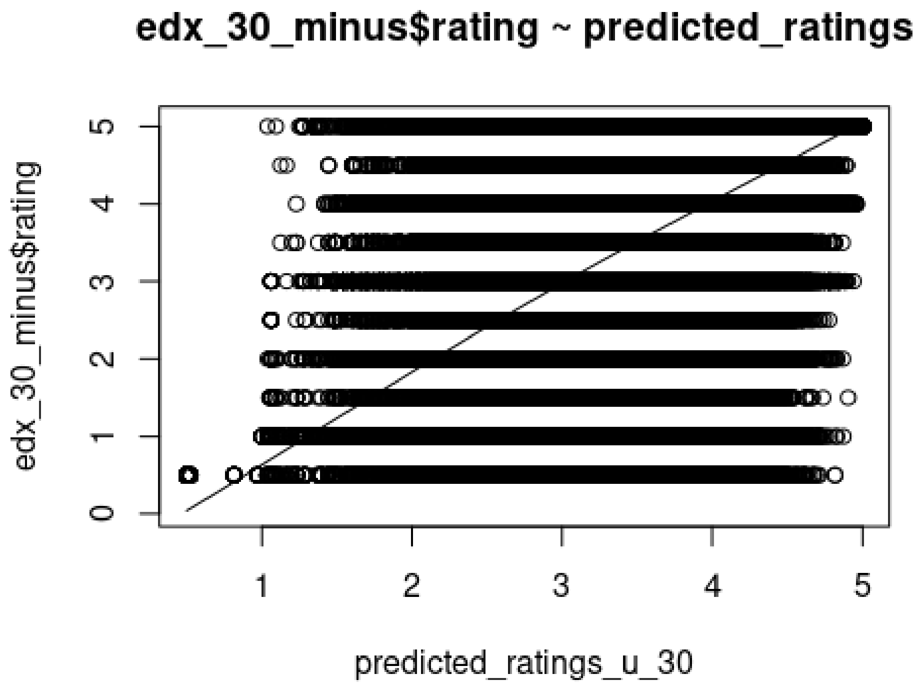

```
#Density plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
plot(density(predicted_ratings_u), main="Density: Predicted ratings",
ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(predicted_ratings_u), 2)))  # density plot for
'Predicted ratings'
polygon(density(predicted_ratings_u), col="red")
plot(density(validation$rating), main="Density Plot: Ratings",
ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(validation$rating), 2)))  # density plot for
'Ratings'
polygon(density(validation$rating), col="red")
```



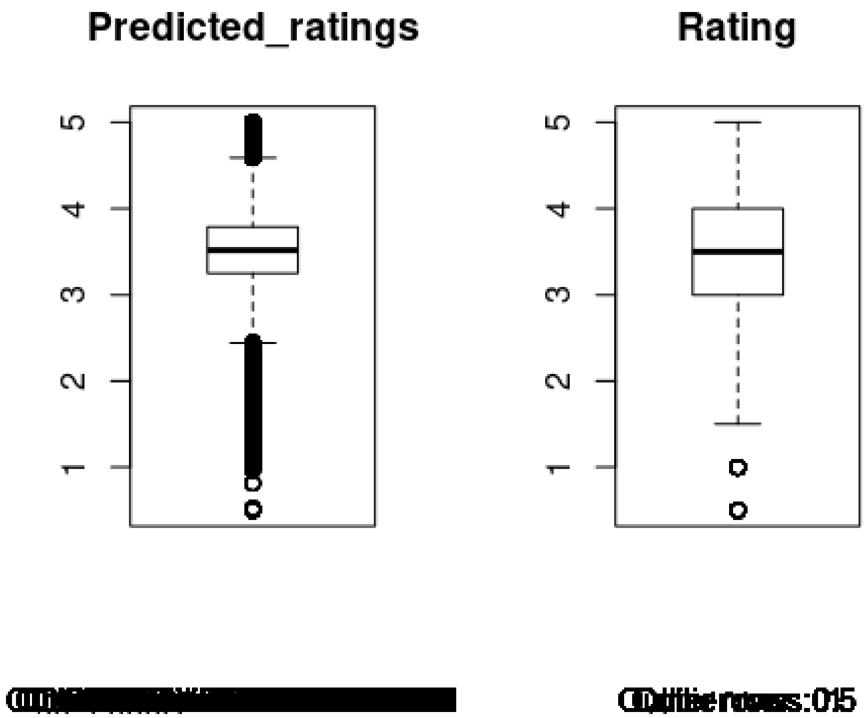

```
# Correlation coefficient
print(cor(predicted_ratings_u, validation$rating))

## [1] 0.3860913
```

Predicted Ratings on edx_30_minus database

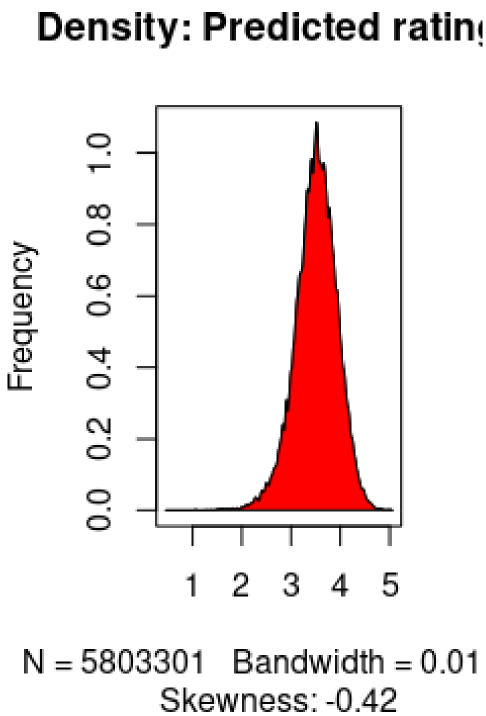
```
## [1] 5 5 5 5 5 5

#Scatter plot
scatter.smooth(x=predicted_ratings_u_30, y=edx_30_minus$rating,
main="edx_30_minus$rating ~ predicted_ratings")
```

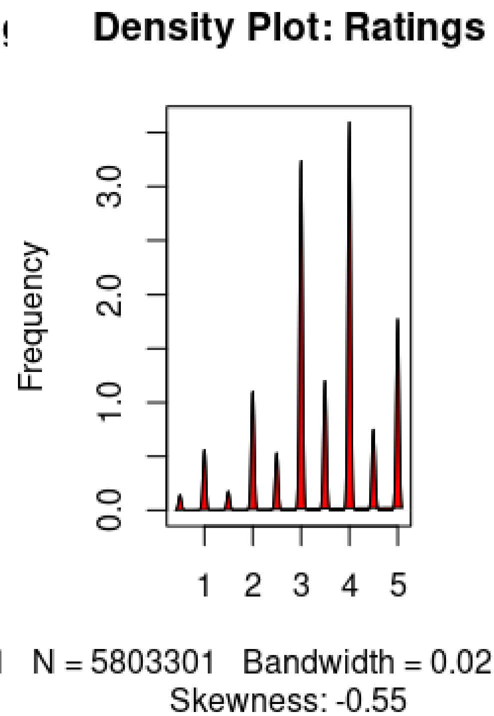



```
#Box plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
boxplot(predicted_ratings_u_30, main="Predicted_ratings",
sub=paste("Outlier rows: ",
boxplot.stats(predicted_ratings_u_30)$out))  # box plot for 'predicted
ratings'
boxplot(edx_30_minus$rating, main="Rating", sub=paste("Outlier rows:
", boxplot.stats(edx_30_minus$rating)$out))  # box plot for
'validation ratings'
```




```
#Density plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
plot(density(predicted_ratings_u_30), main="Density: Predicted
ratings", ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(predicted_ratings_u_30), 2)))  # density plot
for 'Predicted ratings'
polygon(density(predicted_ratings_u_30), col="red")
plot(density(edx_30_minus$rating), main="Density Plot: Ratings",
ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(edx_30_minus$rating), 2)))  # density plot for
'Ratings'
polygon(density(edx_30_minus$rating), col="red")
```






```
# Correlation coefficient
print(cor(predicted_ratings_u_30, edx_30_minus$rating))

## [1] 0.39556
```

User Model

On validation database

```
##
## Call:
## lm(formula = predicted_ratings_u ~ validation$rating, data =
validation)
##
## Coefficients:
##       (Intercept)  validation$rating
##            2.9646             0.1561
```

Linear model formula for User Model on validation database
predicted_ratings_m <- 2.9613 + 0.1568 * validation$rating

On edx_30_minus database

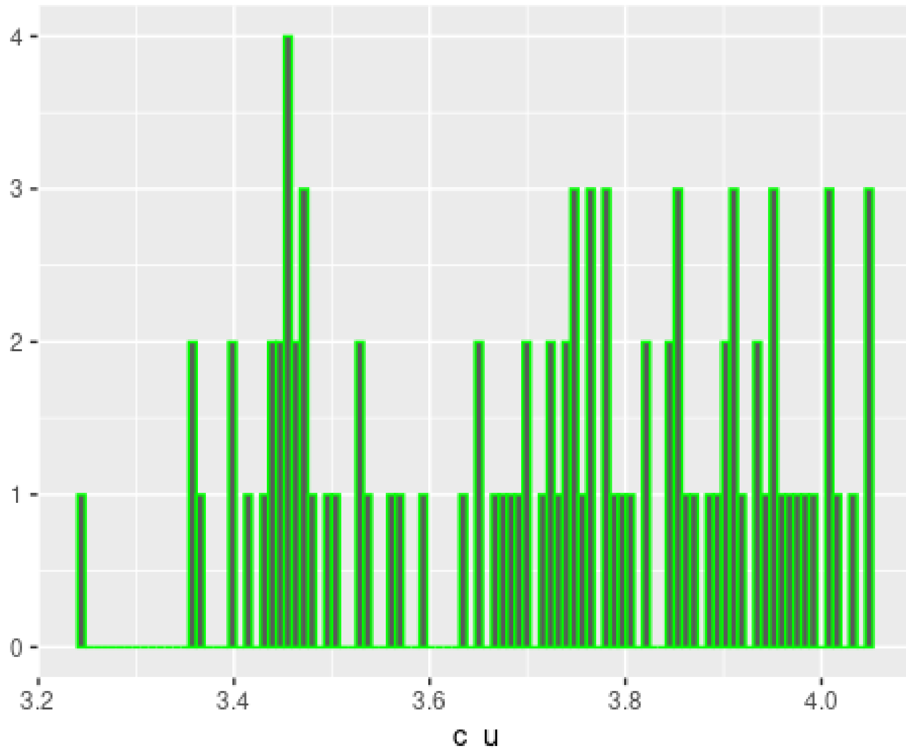
```
##
## Call:
## lm(formula = predicted_ratings_u_30 ~ edx_30_minus$rating, data =
edx_30_minus)
##
```

```
## Coefficients:
##         (Intercept)  edx_30_minus$rating
##              2.9630               0.1568
```

Linear model formula for User Model on edx_30_minus database
predicted_ratings_m <- 2.9613 + 0.1572 * edx_30_minun$rating

4.4 Age effect model

```
## `summarise()` ungrouping output (override with `.groups` argument)
```



Predicted ratings on validation database

Predicted ratings on edx_30_minus database

Age model analyses on validation database

```
# Scatter plot
par(mar=c(1,1,1,1))
graphics.off()
system.time(smoothScatter(predicted_ratings_a, validation$rating, nbin
= 100))  ## 3.3 seconds

##    user  system elapsed
##   0.508   0.016   0.524

#Box plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
boxplot(predicted_ratings_a, main="Predicted_ratings",
```

```
sub=paste("Outlier rows: ", boxplot.stats(predicted_ratings_a)$out))
# box plot for 'predicted_ratings'
boxplot(validation$rating, main="Rating", sub=paste("Outlier rows: ",
boxplot.stats(validation$rating)$out))  # box plot for
'validation$rating'


#Density plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
plot(density(predicted_ratings_a), main="Density: Predicted ratings",
ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(predicted_ratings_a), 2)))  # density plot for
'Predicted ratings'
polygon(density(predicted_ratings_a), col="red")
plot(density(validation$rating), main="Density Plot: Ratings",
ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(validation$rating), 2)))  # density plot for
'Ratings'
polygon(density(validation$rating), col="red")

# Correlation coefficient
print(cor(predicted_ratings_a, validation$rating))

## [1] 0.1443958
```

Age model analyses on edx_30_minus database

```
# Scatter plot
par(mar=c(1,1,1,1))
graphics.off()
system.time(smoothScatter(predicted_ratings_a_30, edx_30_minus$rating,
nbin = 100))  ## 3.3 seconds

##    user  system elapsed
##   2.650   0.328   2.978

#Box plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
boxplot(predicted_ratings_a_30, main="Predicted_ratings",
sub=paste("Outlier rows: ",
boxplot.stats(predicted_ratings_a_30)$out))  # box plot for 'predicted
ratings'
boxplot(edx_30_minus$rating, main="Raiting", sub=paste("Outlier rows:
", boxplot.stats(edx_30_minus$rating)$out))  # box plot for 'ratings'


#Density plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
plot(density(predicted_ratings_a_30), main="Density: Predicted
ratings", ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(predicted_ratings_a_30), 2)))  # density plot
```

```r
for 'Predicted ratings'
polygon(density(predicted_ratings_a_30), col="red")
plot(density(edx_30_minus$rating), main="Density Plot: Ratings",
ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(edx_30_minus$rating), 2)))  # density plot for
'Ratings'
polygon(density(edx_30_minus$rating), col="red")

# Correlation coefficient
print(cor(predicted_ratings_a_30, edx_30_minus$rating))

## [1] 0.04442918
```

On validation database

```
##
## Call:
## lm(formula = predicted_ratings_a ~ validation$rating, data =
validation)
##
## Coefficients:
##       (Intercept)  validation$rating
##           3.43967            0.02076
```

Linear model formula for Age model on validation database
predicted_ratings_m <- 3.43960 + 0.02081 * validation$rating

On edx_30_minus database

```
##
## Call:
## lm(formula = predicted_ratings_a_30 ~ edx_30_minus$rating, data =
edx_30_minus)
##
## Coefficients:
##         (Intercept)  edx_30_minus$rating
##            3.432522             0.001974
```

Linear model formula for Age model on edx_30_minus database
predicted_ratings_m <- 3.435477 + 0.002405 * validation$rating

4.5 Movie and user effect model

Predicted ratings on validation database

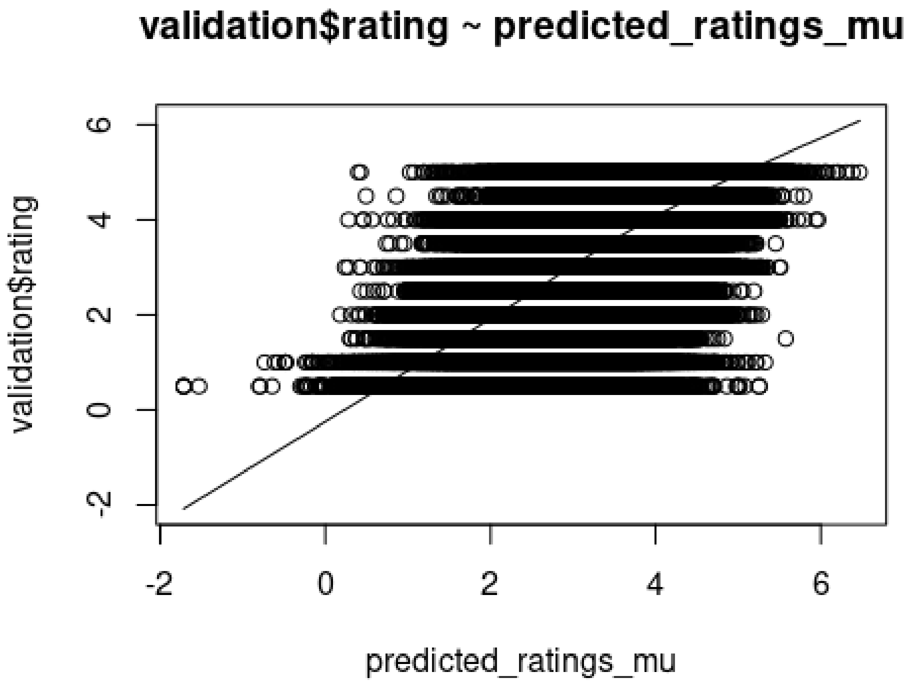
```
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
```

Predicted ratings on edx_30_minus database

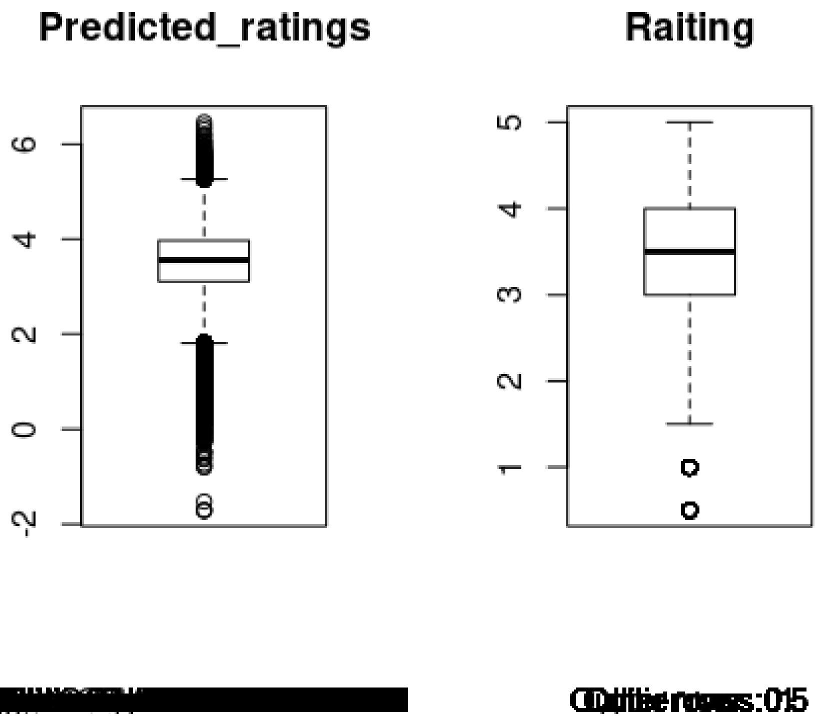
```
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
```

Movie and User Analyze Model on validation database

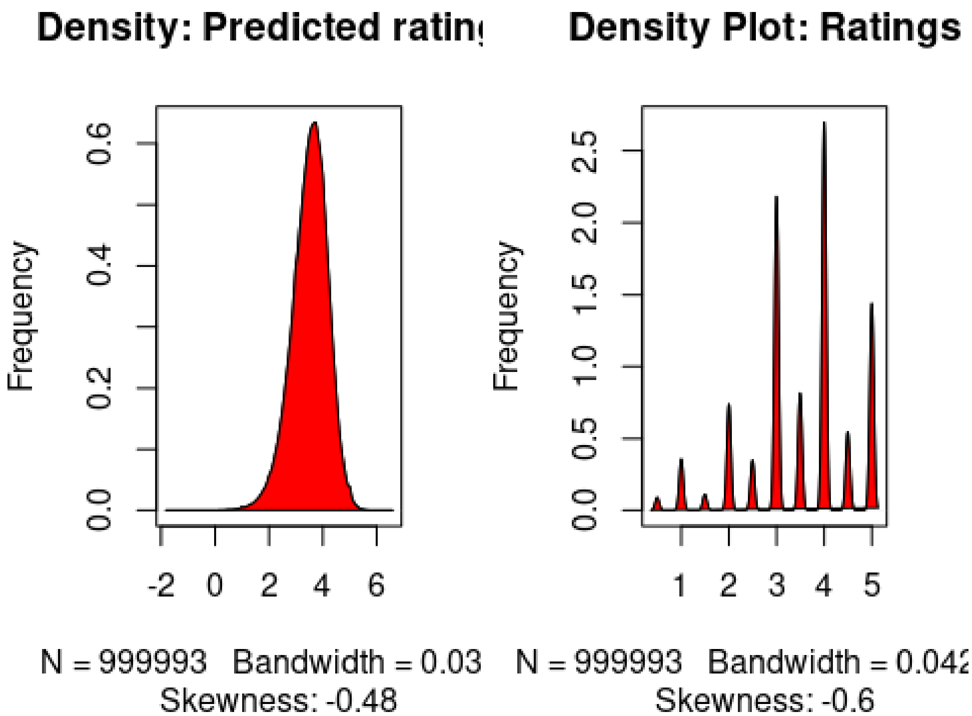
```
# Scatter plot
scatter.smooth(x=predicted_ratings_mu, y=validation$rating,
main="validation$rating ~ predicted_ratings_mu")
```




```
# Box plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
boxplot(predicted_ratings_mu, main="Predicted_ratings",
sub=paste("Outlier rows: ", boxplot.stats(predicted_ratings_mu)$out))
# box plot for 'predicted ratings'
boxplot(edx_30_minus$rating, main="Raiting", sub=paste("Outlier rows:
", boxplot.stats(edx_30_minus$rating)$out))  # box plot for 'rating'
```



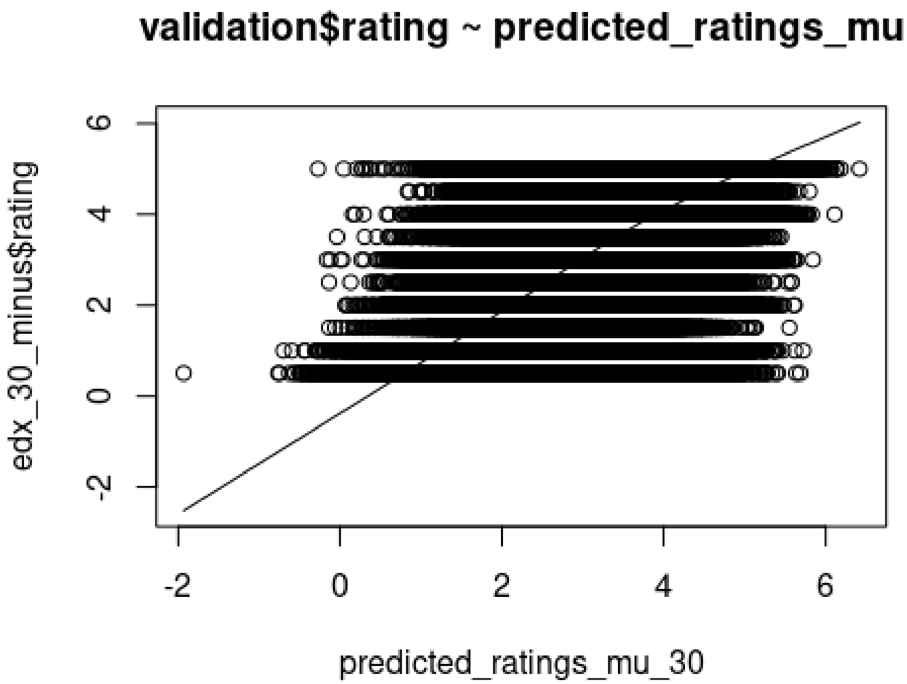

```r
# Density plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
plot(density(predicted_ratings_mu), main="Density: Predicted ratings",
ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(predicted_ratings_mu), 2)))  # density plot for
'Predicted ratings'
polygon(density(predicted_ratings_mu), col="red")
plot(density(validation$rating), main="Density Plot: Ratings",
ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(validation$rating), 2)))  # density plot for
'Ratings'
polygon(density(validation$rating), col="red")
```



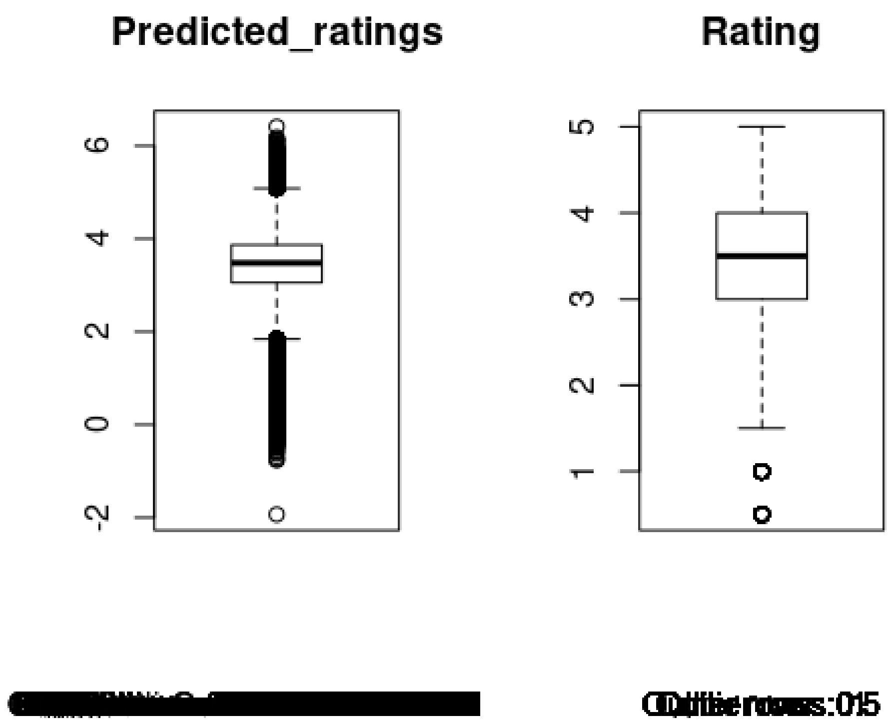

```r
## Print Correlation factor
print(cor(predicted_ratings_mu, validation$rating))

## [1] 0.6281402
```

Movie and User Analyse Model on edx_30_minus database

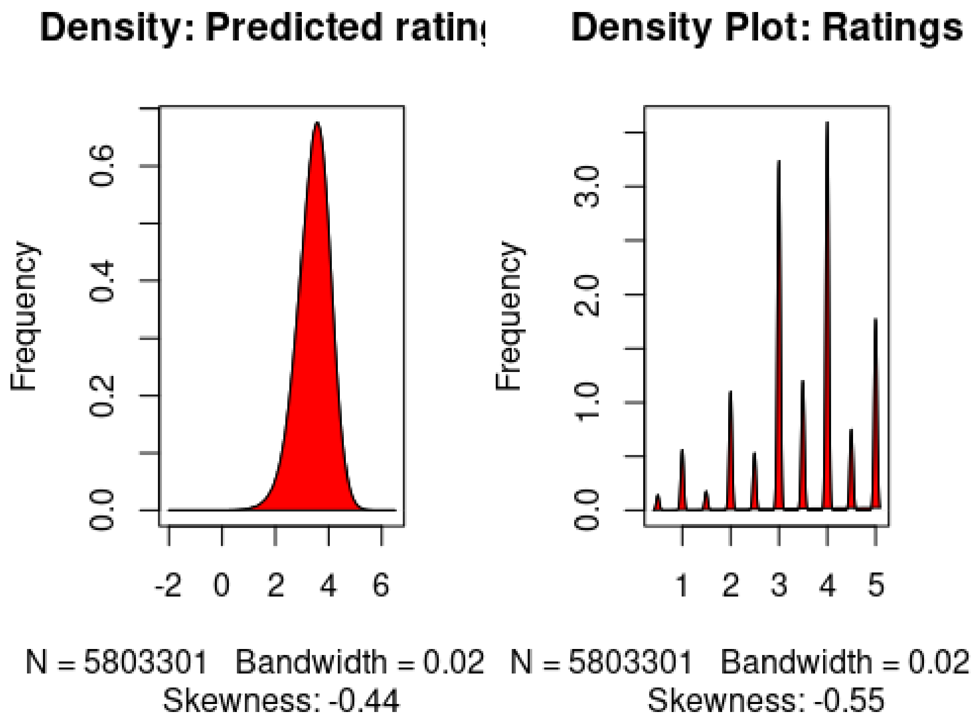
```r
# Scatter plot
scatter.smooth(x=predicted_ratings_mu_30, y=edx_30_minus$rating,
main="validation$rating ~ predicted_ratings_mu")
```




```
# Box plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
boxplot(predicted_ratings_mu_30, main="Predicted_ratings",
sub=paste("Outlier rows: ",
boxplot.stats(predicted_ratings_mu_30)$out))  # box plot for
'predicted ratings'
boxplot(edx_30_minus$rating, main="Rating", sub=paste("Outlier rows:
", boxplot.stats(edx_30_minus$rating)$out))  # box plot for 'rating'
```




```r
# Density plot
par(mfrow=c(1, 2))  # divide graph area in 2 columns
plot(density(predicted_ratings_mu_30), main="Density: Predicted
ratings", ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(predicted_ratings_mu_30), 2)))  # density plot
for 'Predicted ratings'
polygon(density(predicted_ratings_mu_30), col="red")
plot(density(edx_30_minus$rating), main="Density Plot: Ratings",
ylab="Frequency", sub=paste("Skewness:",
round(e1071::skewness(edx_30_minus$rating), 2)))  # density plot for
'Ratings'
polygon(density(edx_30_minus$rating), col="red")
```




```
print(cor(predicted_ratings_mu_30, edx_30_minus$rating))

## [1] 0.5900074
```

Linear model formula for Movie User

On validation database

```
##
## Call:
## lm(formula = predicted_ratings_mu ~ validation$rating, data =
validation)
##
## Coefficients:
##       (Intercept)  validation$rating
##            2.1320             0.3929
```

Linear model formula for Movie User model on validation database predicted_ratings_m <- 2.2249 + 0.3664 * validation$rating

On edx_30_minus database

```
##
## Call:
## lm(formula = predicted_ratings_mu_30 ~ edx_30_minus$rating, data =
edx_30_minus)
##
## Coefficients:
```

```
##         (Intercept)  edx_30_minus$rating
##              2.2578               0.3435
```

Linear model formula for Movie User model on validation database
predicted_ratings_m <- 2.205 + 0.357 * validation$rating

5. RMSE

5.1 Residual Mean Square Error(RMSE)

We will use Residual Mean Square Error(RMSE) to measure accuracy and the typical error of the model. We define a function for RMSE as follow:

```
## Define Residual Mean Square Error(RMSE) function as follow:
RMSE <- function(actual, predicted){
   sqrt(mean((actual - predicted)^2))

 }
```

RMSE for Mean effect model:

On validation database

```
## [1] 1.060651
```

We save the data in a table named final_results which will be used for conclusions

```
## # A tibble: 1 x 3
##   method_used  RMSE database
##   <chr>       <dbl> <chr>
## 1 Average      1.06 validation
```

On edx_30_minus

```
## [1] 1.069492
```

Save the data into final_results

```
## # A tibble: 2 x 3
##   method_used  RMSE database
##   <chr>       <dbl> <chr>
## 1 Average      1.06 validation
## 2 Average      1.07 edx_30_minus
```

5.2 RMSE for Movie effect model :

On validation database

```
## [1] 0.9437046
```

Save data in final_results

```
## # A tibble: 3 x 3
##   method_used  RMSE database
##   <chr>       <dbl> <chr>
## 1 Average     1.06  validation
## 2 Average     1.07  edx_30_minus
## 3 Movie efect 0.944 validation
```

On edx_30_minus database

```
## [1] 0.9550224
```

Save data in final_results

```
## # A tibble: 4 x 3
##   method_used  RMSE database
##   <chr>       <dbl> <chr>
## 1 Average     1.06  validation
## 2 Average     1.07  edx_30_minus
## 3 Movie efect 0.944 validation
## 4 Movie efect 0.955 edx_30_minus
```

5.3 RMSE for User effect model

On validation database

```
## [1] 0.9785992
```

Save data in final_results

```
## # A tibble: 5 x 3
##   method_used  RMSE database
##   <chr>       <dbl> <chr>
## 1 Average     1.06  validation
## 2 Average     1.07  edx_30_minus
## 3 Movie efect 0.944 validation
## 4 Movie efect 0.955 edx_30_minus
## 5 User efect  0.979 validation
```

On edx_30_minus database

```
## [1] 0.9819874
```

Save data in final_results

```
## # A tibble: 6 x 3
##   method_used  RMSE database
##   <chr>       <dbl> <chr>
## 1 Average     1.06  validation
## 2 Average     1.07  edx_30_minus
## 3 Movie efect 0.944 validation
## 4 Movie efect 0.955 edx_30_minus
## 5 User efect  0.979 validation
## 6 User efect  0.982 edx_30_minus
```

5.4 RMSE for Age effect model

On validation database

```
## [1] 1.049535
```

Save data in final_results

```
## # A tibble: 7 x 3
##   method_used  RMSE database
##   <chr>       <dbl> <chr>
## 1 Average     1.06  validation
## 2 Average     1.07  edx_30_minus
## 3 Movie efect 0.944 validation
## 4 Movie efect 0.955 edx_30_minus
## 5 User efect  0.979 validation
## 6 User efect  0.982 edx_30_minus
## 7 Age efect   1.05  validation
```

On edx_30_minus database

```
## [1] 1.065934
```

Save data in final_results

```
## # A tibble: 8 x 3
##   method_used  RMSE database
##   <chr>       <dbl> <chr>
## 1 Average     1.06  validation
## 2 Average     1.07  edx_30_minus
## 3 Movie efect 0.944 validation
## 4 Movie efect 0.955 edx_30_minus
## 5 User efect  0.979 validation
## 6 User efect  0.982 edx_30_minus
## 7 Age efect   1.05  validation
## 8 Age efect   1.07  edx_30_minus
```

4.5 RMSE for Movie and user effect model

On validation database

```
## [1] 0.8252969
```

Save data in final results

```
## # A tibble: 9 x 3
##   method_used            RMSE database
##   <chr>                 <dbl> <chr>
## 1 Average               1.06  validation
## 2 Average               1.07  edx_30_minus
## 3 Movie efect           0.944 validation
## 4 Movie efect           0.955 edx_30_minus
## 5 User efect            0.979 validation
## 6 User efect            0.982 edx_30_minus
```

```
## 7 Age efect            1.05  validation
## 8 Age efect            1.07  edx_30_minus
## 9 Movie and user efect 0.825 validation
```

On edx_30_minus database

```
## [1] 0.8615231
```

Save data in final results

```
## # A tibble: 10 x 3
##    method_used           RMSE database
##    <chr>                <dbl> <chr>
##  1 Average              1.06  validation
##  2 Average              1.07  edx_30_minus
##  3 Movie efect          0.944 validation
##  4 Movie efect          0.955 edx_30_minus
##  5 User efect           0.979 validation
##  6 User efect           0.982 edx_30_minus
##  7 Age efect            1.05  validation
##  8 Age efect            1.07  edx_30_minus
##  9 Movie and user efect 0.825 validation
## 10 Movie and user efect 0.862 edx_30_minus

##
## Call:
## lm(formula = validation$rating ~ predicted_ratings_mu +
predicted_ratings_m,
##     data = validation)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7601 -0.4728  0.0538  0.5459  4.6100
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -0.050614   0.006013  -8.418   <2e-16 ***
## predicted_ratings_mu  0.992649   0.001790 554.649   <2e-16 ***
## predicted_ratings_m   0.021763   0.002440   8.921   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8253 on 999990 degrees of freedom
## Multiple R-squared:  0.3946, Adjusted R-squared:  0.3946
## F-statistic: 3.259e+05 on 2 and 999990 DF,  p-value: < 2.2e-16
```

6. Conclusion

We saw that the lowest RMSE is from Movie User model 0.8251770 which show us that is the best model to use in predictions of the new ratings.