Capstone Project - HarvardX: PH125.9x Data Science

Imad Belkheiri

19/08/2020

Find this project online at: https://github.com/ImadBM/Capstone-MovieLens-Project

1- Introduction

This document presents the findings and methodology used in the capstone project of the Data Science certificate (PH125.9x) offered by HarvardX (https://vpal.harvard.edu/harvardx).

The subject studied in this capstone project is about creating a movie recommendation system. Wikipedia defines recommendation systems as follows: "A recommender system, or a recommendation system (sometimes replacing 'system' with a synonym such as platform or engine), is a subclass of information filtering system that seeks to predict the"rating" or "preference" a user would give to an item"

Considering that the world is and becoming busier day after day, recommendation systems have become extremely popular and important into attracting and satisfying customers. Customers' time optimization in consuming movies, as example, is especially important into offering a unique, personalized and satisfying experience.

Netflix company has organized an open competition for the development of a movie rating algorithm based only on ratings. The prize of 1 million USD was won on September 21, 2009 by the team "BellKor's Pragmatic Chaos".

For this project, the MovieLens dataset will be used to create the perdition model. We will use the 10 million data entries to construct the testing and validation sets. In the other hand, the Residual Mean Squared Error (RMSE) method has been chosen to evaluate the accuracy of the prediction model.

In this document we will go through 3 main parts of data collection and processing, data exploration and analysis, model deployment and optimization.

2- Environment Configuration

Optimisation of working environment will be achieved through different parts:

- Creation of specific data management folder
- Testing of existence of downloaded files
- Storage of processed and cleaned data frames for reusability (optimization of code testing time)

```r
#We want to create a specific folder of files storages, if not existed
if (file.exists("Data") == FALSE){
  dir.create("Data")
}

#We want to download the data file, if not already downloaded
if(file.exists('Data/ml-10m.zip') == FALSE) {
  download.file("http://files.grouplens.org/datasets/movielens/ml-10m.zip",
                'Data/ml-10m.zip')
}
```

```r
#For computer time optimization, we want to store the data set after processing and cleaning in the com
#Different methods are available, lets test execution time and file size for each method


#function to calculate time spent and file size with files extensions, storage functions' names
#and method names as variables
storage_compare <- function(x,a){
  time_results <- data.frame()
  for (i in 1:dim(a)[1]){
    time_spent <- system.time(get(a$storage_functions[i])(x,a$storage_files[i])) #execute the function
    time_results <- bind_rows(time_results,data_frame(filetype = a$storage_modes[i], #add results to da
                user.self = time_spent[1], sys.self = time_spent[2], elapsed = time_spent[3],
                size = humanReadable(file.info(a$storage_files[i])$size)))
  }
  time_results
}

storage_files <- c("Data/ml-10M100K/ratings.rda","Data/ml-10M100K/ratings.cvs", #file extension name
                "Data/ml-10M100K/ratings.feather","Data/ml-10M100K/ratings.fst")
storage_modes <- c("RDS", "CVS", "Feather", "FST") #method names
storage_functions <- c("saveRDS", "vroom_write", "write_feather", "write_fst") #functions to be called
storage_variables <- data.frame(storage_modes,storage_files,storage_functions)

ratings <- fread(text = gsub("::", "\t", readLines(unzip('Data/ml-10m.zip',
                "ml-10M100K/ratings.dat",exdir = "Data"))),
                col.names = c("userId", "movieId", "rating", "timestamp"))

storage_compare_result <- storage_compare(ratings,storage_variables)
storage_compare_result
```

```
##   filetype user.self sys.self elapsed      size
## 1      RDS      8.17     0.14    8.45  44.6 MiB
## 2      CVS      2.93     0.35    1.00 224.2 MiB
## 3  Feather      0.08     0.15    0.71 190.7 MiB
## 4      FST      0.24     0.05    0.10  78.0 MiB
```

In the process of data storage optimization, two important variables are considered: size and execution time. The best execution time is achieved with FST method with 0.09 minutes and the best minimum size of 44.6 MiB is achieved by RDS method.

Considering the relative availability of storage space, the FST method is recommended as it enables to run multiple times the code for testing with minimum execution time.

3- Data Collection

```r
#Lets unzip and download data sets to create a movielens object
if(file.exists("Data/ml-10M100K/movielens.fst") == TRUE){
  movielens <- read_fst("Data/ml-10M100K/movielens.fst") %>% data.table()
} else {
  ratings <- fread(text = gsub("::", "\t", readLines(unzip('Data/ml-10m.zip',
                  "ml-10M100K/ratings.dat",exdir = "Data"))),
                   col.names = c("userId", "movieId", "rating", "timestamp"))

  movies <- str_split_fixed(readLines(unzip('Data/ml-10m.zip', "ml-10M100K/movies.dat",
                  exdir = "Data")), "\\::", 3)

  colnames(movies) <- c("movieId", "title", "genres")

  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")

  #Save data frame for future use
  write_fst(movielens,"Data/ml-10M100K/movielens.fst")

  #cleaning of remove files and variables
  file.remove("Data/ml-10M100K/ratings.dat","Data/ml-10M100K/movies.dat")
  rm(ratings,movies)
}
```

4- Data Processing

Movilens data frame head lines:

```r
str(movielens)
```

```
## 'data.frame':    10000054 obs. of  6 variables:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ movieId  : num  122 185 231 292 316 329 355 356 362 364 ...
##  $ rating   : num  5 5 5 5 5 5 5 5 5 5 ...
##  $ timestamp: int  838985046 838983525 838983392 838983421 838983392 838983392 838984474 838983653 83
##  $ 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"
```

Movielens data frame contains variables:

- userId : a unique identifier for users rating the movies
- movieId : a unique identifier for a movie
- rating : the rate given by a user to a movie

- timestamp : POSIXct objects defining the date and time the rate was given
- title : movie title
- genres : movie categories regrouped

Movielens data frame's variables can be processed to enhance information usage:

```r
#Lets extract the movie rating year from timestamp
if (length(which(names(movielens) == "year_rating")) == 0) {
  movielens <- movielens %>% mutate(timestamp = anytime(timestamp))
  movielens <- movielens %>% mutate(year_rating = year(timestamp))
}

#Lets split title into movie title and movie production year
if (length(which(names(movielens) == "year_movie")) == 0) {
  movielens <- movielens %>% mutate(year_movie = str_extract(title,
                   "[0-9][0-9][0-9][0-9]\\)$")) %>% mutate(year_movie =
                   sub("\\)$", "", year_movie))
  movielens$year_movie <- as.integer(movielens$year_movie)
}

if (length(which(names(movielens) == "movie_title")) == 0) {
  movielens <- movielens %>% mutate(movie_title = sub(" \\([0-9]+\\)$", "", title))
}
```

```r
#Save data frame for future use
write_fst(movielens,"Data/ml-10M100K/movielens.fst")
```

5- Data Exploration and Analysis

We can see that the new Movielens data frame contains a total of 10 000 054 ratings and 9 variables

```r
dim(movielens)
```

```
## [1] 10000054        9
```

Movilens data frame new head lines:

```r
str(movielens)
```

```
## 'data.frame':    10000054 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 ...
##  $ timestamp  : POSIXct, format: "1996-08-02 11:24:06" "1996-08-02 10:58:45" ...
##  $ 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
##  $ year_rating: int  1996 1996 1996 1996 1996 1996 1996 1996 1996 1996 ...
##  $ year_movie : int  1992 1995 1994 1995 1994 1994 1994 1994 1994 1994 ...
##  $ movie_title: chr  "Boomerang" "Net, The" "Dumb & Dumber" "Outbreak" ...
```

5.1- UserID

We can see that the data frame contains a total of 69 878 uniq rating users

```
n_distinct(movielens$userId)
```

```
## [1] 69878
```

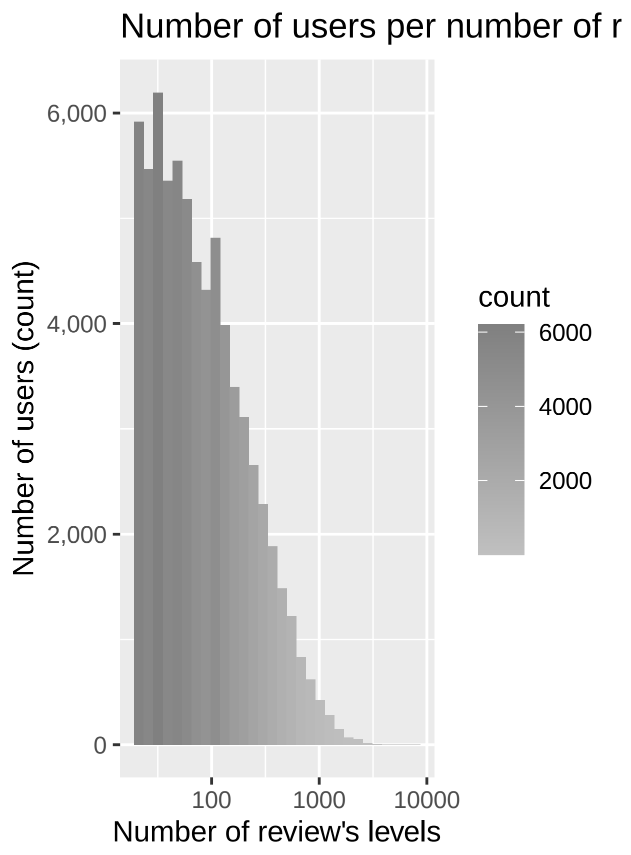
The list doesnt contains any NAs

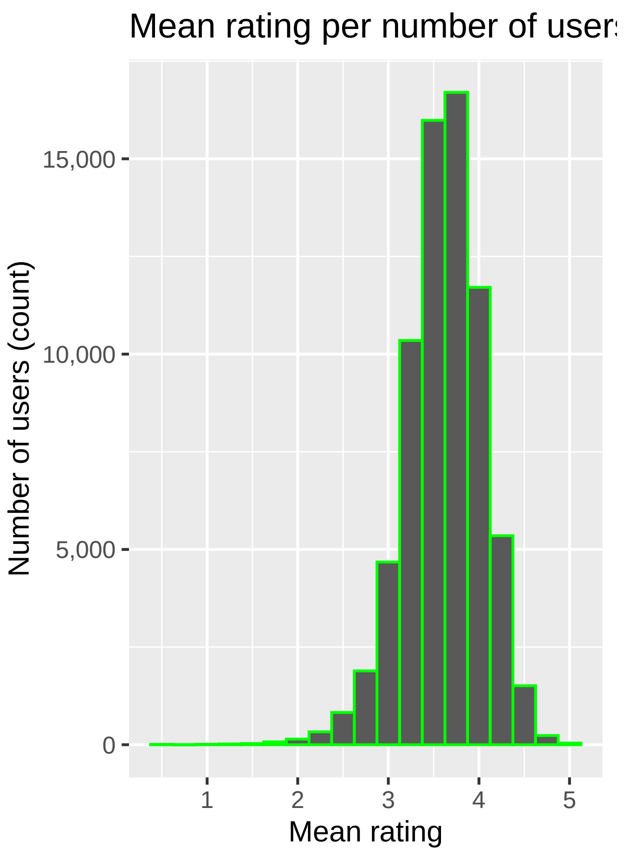
```
sum(is.na(movielens$userId) == TRUE)
```

```
## [1] 0
```

```
#Lets analyse how many reviews are done by unique users
p1 <- movielens %>% group_by(userId) %>% summarise(n = n()) %>%
  ggplot(aes(n)) +
  geom_histogram(aes(fill=..count..)) +
  scale_x_log10() +
  scale_fill_gradient(low="#C0C0C0", high="#808080") +
  scale_y_continuous(labels = comma) +
  labs(title="Number of users per number of review' levels",y= "Number of users (count)", x = "Number o

#Lets analyse what is the mean rating level by unique users
p2 <- movielens %>% group_by(userId) %>%
  summarise(mean_rating = mean(rating)) %>%
  ggplot(aes(mean_rating)) +
  geom_histogram(binwidth = 0.25, color = "green") +
  scale_y_continuous(labels = comma) +
  labs(title="Mean rating per number of users' levels",y= "Number of users (count)", x = "Mean rating")

grid.arrange(p1, p2, ncol=2)
```

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





In the first graph, we can notice that most users are doing less than 100 reviews per user which is totally logic as users are supposed to watch the movies before rating them (watching more than 100 movies is very time consuming)

In the other hand, the second graph inform us that majority of users have rated movies in average between 3 and 4. What about low averages means, we need to analyse the rates in order to have a clearer vision.

5.2- Ratings

We can see that the data frame contains a total of 10 uniq rating levels

```
n_distinct(movielens$rating)
```

```
## [1] 10
```

With 10 levels ranging from 0.5 to 10 (by 0.5 levels)

```
as.factor(movielens$rating) %>% levels()
```

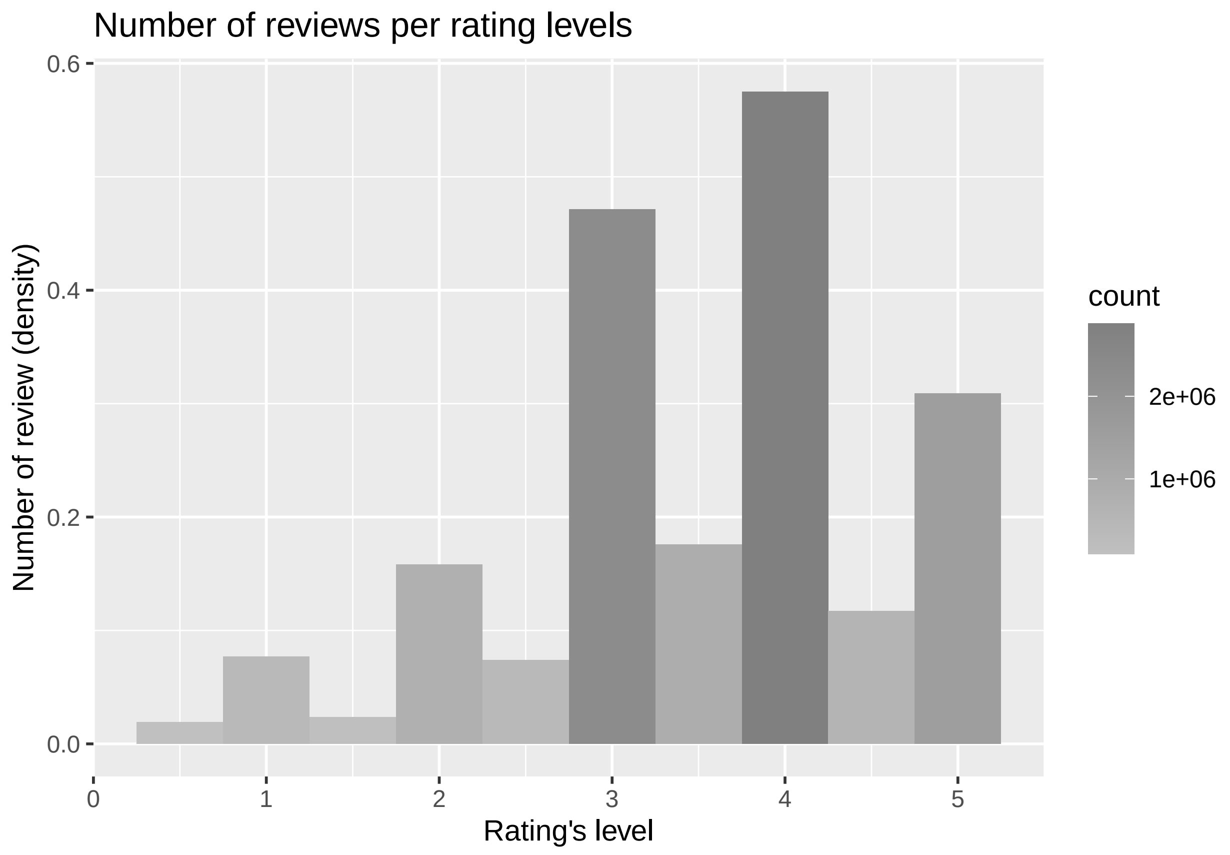
```
##  [1] "0.5" "1"   "1.5" "2"   "2.5" "3"   "3.5" "4"   "4.5" "5"
```

The list doesnt contains any NAs

```r
sum(is.na(movielens$rating) == TRUE)
```

```
## [1] 0
```

```r
#Lets analyse the number of reviews by rating level
movielens %>% ggplot(aes(rating)) +
  geom_histogram(aes(y=..density.., fill=..count..), binwidth=0.5) +
  scale_fill_gradient(low="#C0C0C0", high="#808080") +
  labs(title="Number of reviews per rating levels",y= "Number of review (density)", x = "Rating's level
```



The rating analysis gives two insights:

- Users tend to rate by integers numbers as 1,2,3..5 and less with digital 0.5,1.5,… –> this questions the necessity of 10 levels rather than 5 integer levels
- Our question raised in section 5.1 is confirmed by this analysis: users rates mainly between 3 and 4

Majority of rating are bigger than 3 which means that majority of movies are well rated. Here we can see a bias that may be explained by the fact that users don't rate bad movies.

5.3- MovieID

We can see that the data frame contains a total of 10 677 uniq movies

```
n_distinct(movielens$movieId)
```

```
## [1] 10677
```

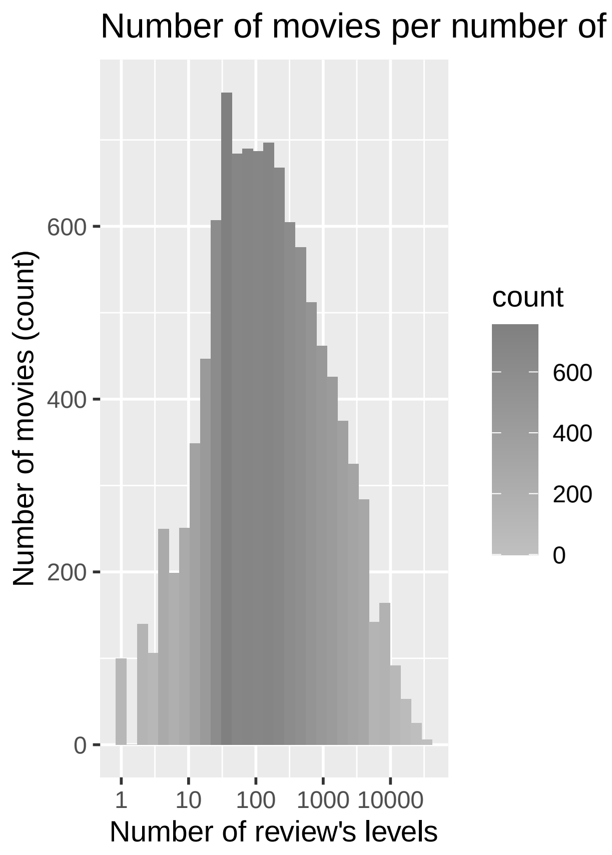
The list doesnt contains any NAs

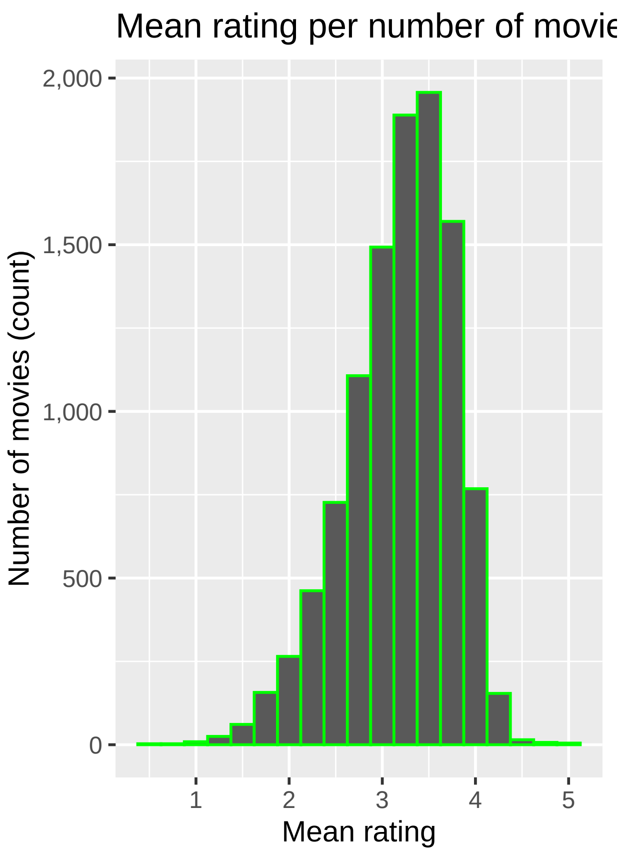
```
sum(is.na(movielens$movieId) == TRUE)
```

```
## [1] 0
```

```
#Lets analyse how many reviews are done by unique movies
p1 <- movielens %>% group_by(movieId) %>% summarise(n = n()) %>%
  ggplot(aes(n)) +
  geom_histogram(aes(fill=..count..)) +
  scale_x_log10() +
  scale_fill_gradient(low="#C0C0C0", high="#808080") +
  scale_y_continuous(labels = comma) +
  labs(title="Number of movies per number of review’ levels",y= "Number of movies (count)", x = "Number

#Lets analyse what is the mean rating level by unique movies
p2 <- movielens %>% group_by(movieId) %>%
  summarise(mean_rating = mean(rating)) %>%
  ggplot(aes(mean_rating)) +
  geom_histogram(binwidth = 0.25, color = "green") +
  scale_y_continuous(labels = comma) +
  labs(title="Mean rating per number of movies’ levels",y= "Number of movies (count)", x = "Mean rating

grid.arrange(p1, p2, ncol=2)
```

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





In the first graph, we can notice that the most movies are having between 10 and 1000 reviews.

In the other hand, the second graph inform us that majority of movies are rated in average between 3 and 4. This confirms our first two analysis.

5.4- movie year

We can see that the data frame contains a total of 94 uniq movies' production year

```
n_distinct(movielens$year_movie)
```

```
## [1] 94
```

The movies rated in the dataset are produced between 1915 and 2008

```
as.factor(movielens$year_movie) %>% levels()
```

```
##  [1] "1915" "1916" "1917" "1918" "1919" "1920" "1921" "1922" "1923" "1924"
## [11] "1925" "1926" "1927" "1928" "1929" "1930" "1931" "1932" "1933" "1934"
## [21] "1935" "1936" "1937" "1938" "1939" "1940" "1941" "1942" "1943" "1944"
## [31] "1945" "1946" "1947" "1948" "1949" "1950" "1951" "1952" "1953" "1954"
## [41] "1955" "1956" "1957" "1958" "1959" "1960" "1961" "1962" "1963" "1964"
## [51] "1965" "1966" "1967" "1968" "1969" "1970" "1971" "1972" "1973" "1974"
## [61] "1975" "1976" "1977" "1978" "1979" "1980" "1981" "1982" "1983" "1984"
## [71] "1985" "1986" "1987" "1988" "1989" "1990" "1991" "1992" "1993" "1994"
## [81] "1995" "1996" "1997" "1998" "1999" "2000" "2001" "2002" "2003" "2004"
## [91] "2005" "2006" "2007" "2008"
```

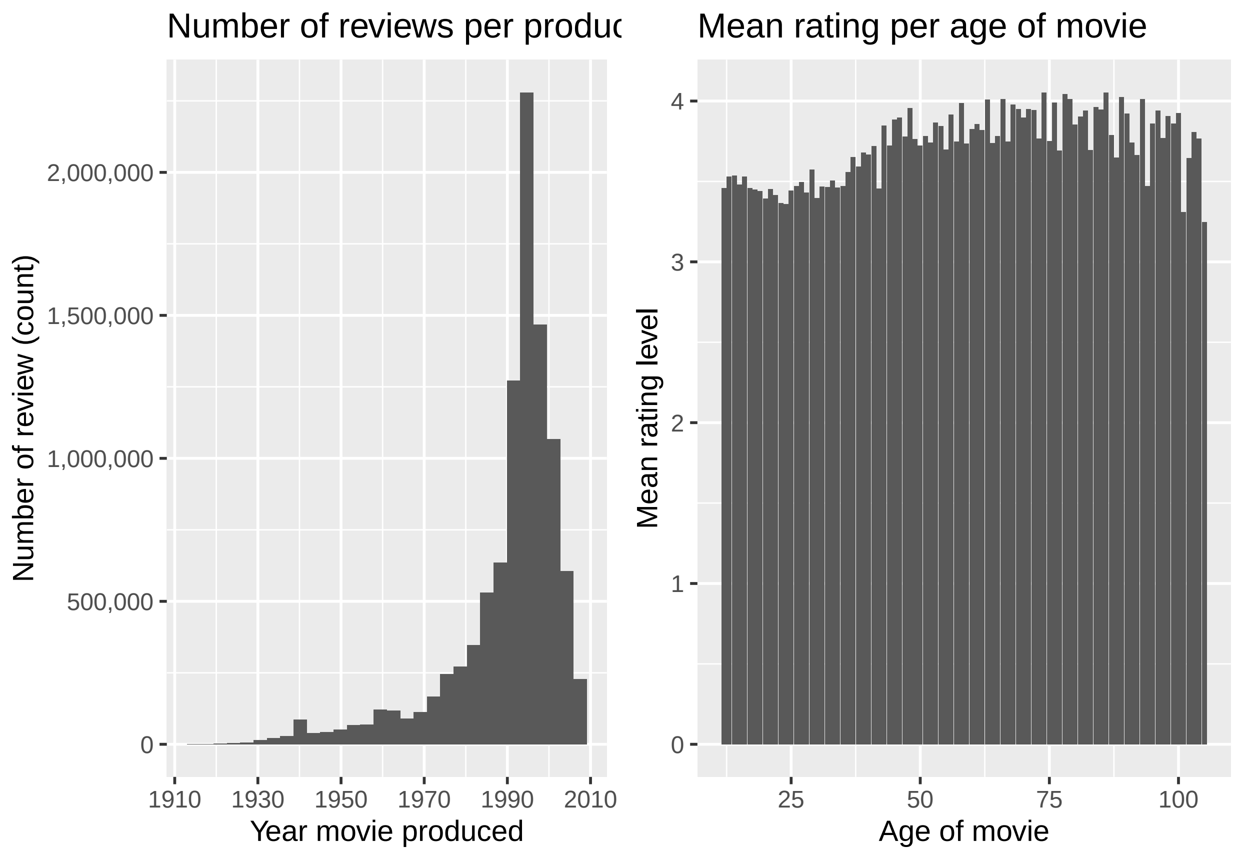
The list doesnt contains any NAs

```
sum(is.na(movielens$year_movie) == TRUE)
```

```
## [1] 0
```

```
#Lets analyse number of movies per year of production
p1 <- movielens %>% ggplot(aes(year_movie)) +
  geom_histogram() +
  scale_y_continuous(labels = comma) +
  labs(title="Number of reviews per production year",y= "Number of review (count)", x = "Year movie proc

#Lets analyse what is the mean rating level by year of production
p2 <- movielens %>% mutate(age_movie = 2020 - year_movie) %>%
  group_by(age_movie) %>%
  summarise(mean_rating = mean(rating)) %>%
  ggplot() +
  geom_bar(aes(x = age_movie, y = mean_rating), stat = "identity") +
  labs(title="Mean rating per age of movie",y= "Mean rating level", x = "Age of movie")

grid.arrange(p1, p2, ncol=2)
```

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



In the first graph, we can notice the relatively recent movies experience to have a greater number of ratings.

In the other hand, the second graph inform us that movies are having relatively similar rating means independently of production year.

5.5- rating year

We can see that the data frame contains a total of 15 uniq rating years

```
n_distinct(movielens$year_rating)
```

```
## [1] 15
```

The movies rated in the dataset are rated between 1995 and 2009

```
as.factor(movielens$year_rating) %>% levels()
```

```
##  [1] "1995" "1996" "1997" "1998" "1999" "2000" "2001" "2002" "2003" "2004"
## [11] "2005" "2006" "2007" "2008" "2009"
```

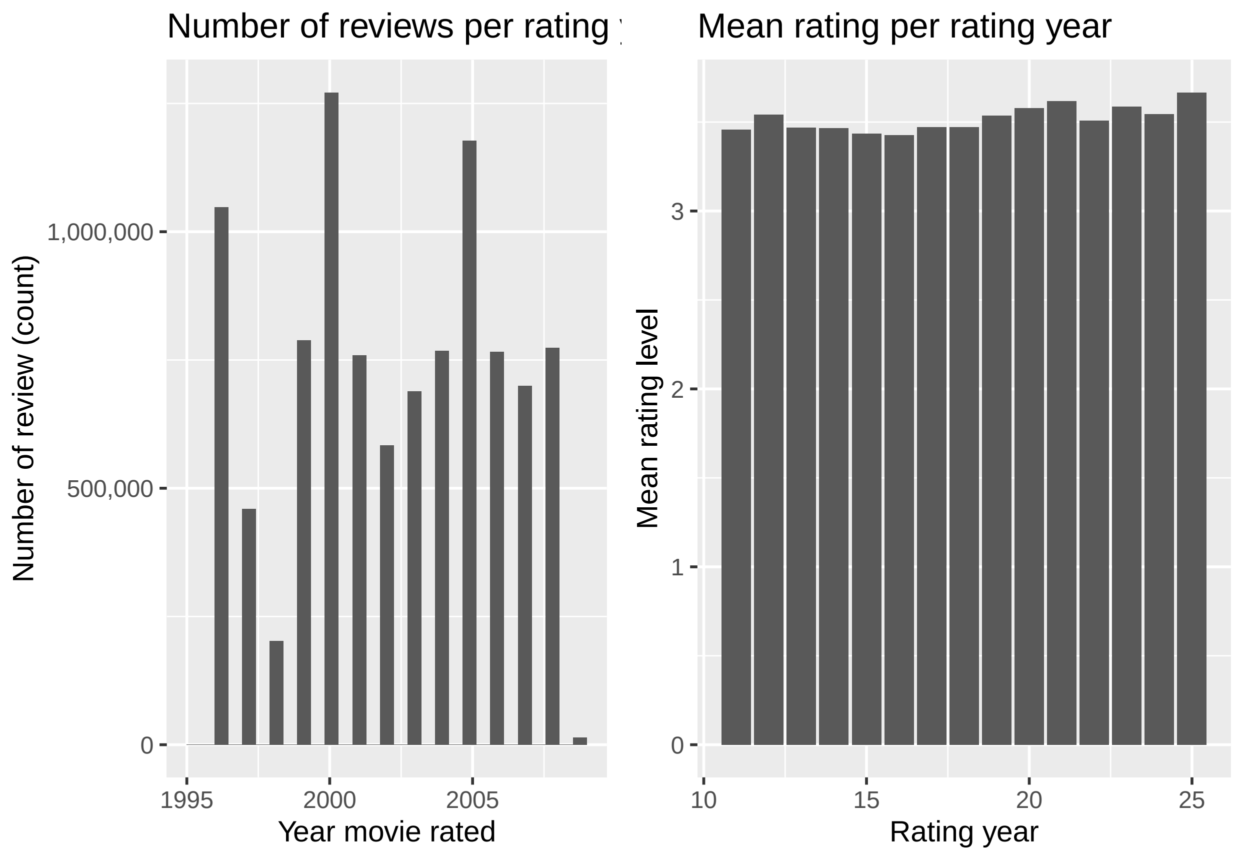
The list doesnt contains any NAs

```
sum(is.na(movielens$year_rating) == TRUE)
```

```
## [1] 0
```

```
#Lets analyse number of movies per year of rating
p1 <- movielens %>% ggplot(aes(year_rating)) +
  geom_histogram() +
  scale_y_continuous(labels = comma) +
  labs(title="Number of reviews per rating year",y= "Number of review (count)", x = "Year movie rated")

#Lets analyse what is the mean rating level by year of rating
p2 <- movielens %>% mutate(age_rating = 2020 - year_rating) %>%
  group_by(age_rating) %>%
  summarise(mean_rating = mean(rating)) %>%
  ggplot(aes(mean_rating)) +
  geom_bar(aes(x = age_rating, y = mean_rating), stat = "identity") +
  labs(title="Mean rating per rating year",y= "Mean rating level", x = "Rating year")

grid.arrange(p1, p2, ncol=2)
```

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



In the first graph, we can notice that no trend is detected a part of some top years which is may be due to marketing efforts for improve number of ratings.

In the other hand, the second graph inform us that movies are having relatively similar rating means independently of rating year.

5.6- rating age

When analyzing rating age levels, we notice that some are less than 0.

```
test <- movielens %>% mutate(rating_age = year_rating - year_movie) %>% mutate(rating_age = as.factor(ra
test$rating_age %>% levels()
```

```
##  [1] "-2" "-1" "0"  "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12"
## [16] "13" "14" "15" "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27"
## [31] "28" "29" "30" "31" "32" "33" "34" "35" "36" "37" "38" "39" "40" "41" "42"
## [46] "43" "44" "45" "46" "47" "48" "49" "50" "51" "52" "53" "54" "55" "56" "57"
## [61] "58" "59" "60" "61" "62" "63" "64" "65" "66" "67" "68" "69" "70" "71" "72"
## [76] "73" "74" "75" "76" "77" "78" "79" "80" "81" "82" "83" "84" "85" "86" "87"
## [91] "88" "89" "90" "91" "92" "93"
```

This is due to some errors in the data set where year of rating is less than year of movie production. Some examples:

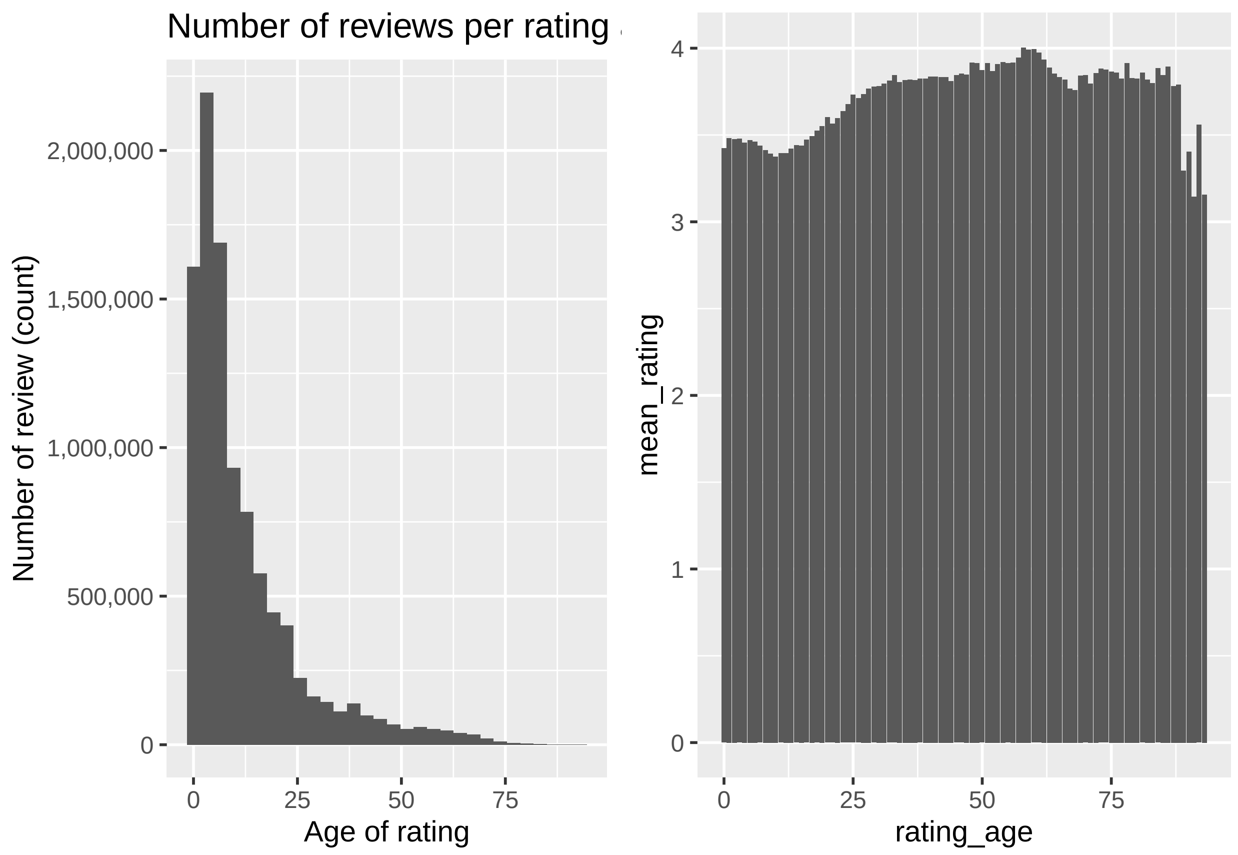
```
movielens[which(test$rating_age == "-2"),] %>% select(userId,movieId,rating,title,timestamp)
```

```
##         userId movieId rating                 title           timestamp
## 1068593   8010     887      3 Talk of Angels (1998) 1996-10-10 08:57:49
## 7492203  53545     887      1 Talk of Angels (1998) 1996-09-17 11:56:51
## 9080775  65117     887      2 Talk of Angels (1998) 1996-09-23 21:49:48
```

```
#Lets clean the data set
movielens <- movielens[-which(test$rating_age == "-1" | test$rating_age == "-2"),]
```

```
#Lets analyse number of movies per age of rating
p1 <- movielens %>% mutate(rating_age = year_rating - year_movie) %>%
  ggplot(aes(rating_age)) +
  geom_histogram() +
  scale_y_continuous(labels = comma) +
  labs(title="Number of reviews per rating age",y= "Number of review (count)", x = "Age of rating")

#Lets analyse what is the mean rating level by age of rating
p2 <- movielens %>% mutate(rating_age = year_rating - year_movie) %>%
  group_by(rating_age) %>%
  summarise(mean_rating = mean(rating)) %>%
  ggplot() +
  geom_bar(aes(x = rating_age, y = mean_rating), stat = "identity")

grid.arrange(p1, p2, ncol=2)
```

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



In the first graph, we can notice that the younger is the rating the more reviews there are which can be explained by the interest of users in recent movies compared with year of rating.

In the other hand, the second graph inform us that movies are having relatively similar rating means independently of age of rating.

```
rm(p1,p2,test) #cleaning of files and variables used
write_fst(movielens,"Data/ml-10M100K/movielens.fst") #Save data frame for future use
```

6- Model preparation

```
#creation of a validation set that will be 10% of MovieLens data
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

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

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

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

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

nrow(movielens) - nrow(edx) - nrow(validation)
```

```
## [1] 0
```

```r
#defining the RMSe function
RMSE <- function(true_ratings, predicted_ratings){
  sqrt(mean((true_ratings - predicted_ratings)^2))
}

#cleaning of files and variables used
rm(temp,test_index,removed,movielens)
```

7- Model deployment

In our modeling we will use the RMSE method and focus on level less than focus on obtaining an RMS <= 0.86499

Level 1 - Mean:

We will approximate the validation ratings by the mean of all rating in the test set :

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

```r
mu_hat <- mean(edx$rating)
level1_rmse <- RMSE(validation$rating, mu_hat)
level1_rmse
```

```
## [1] 1.060789
```

```r
rmse_conclusions <- data_frame(method = "Level1 - Mean:", RMSE = level1_rmse)
```

Level 2 - Movie Bias:

From our previous analysis 5.4 many movies are rated differently. We can take as example the movies that are graded closely after production are rated by more users.

We can improve our model by including the movies effect:

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

```r
#first, estimate the effect of each movie bi
mu <- mean(edx$rating)

movie_avgs <- edx %>%
  group_by(movieId) %>%
  summarise(b_i = mean(rating - mu))

# then, make the prediction using the new model y= mu + b
```

```
predicted_ratings <- mu + validation %>%
  left_join(movie_avgs, by='movieId') %>%
  pull(b_i)

level2_rmse <- RMSE(predicted_ratings, validation$rating)
level2_rmse
```

```
## [1] 0.9439936
```

```
# Add the rmse result of the new model to the rmse data frame
rmse_conclusions <- bind_rows(rmse_conclusions,
                          data_frame(method="Level2 - Movie Bias:",
                                     RMSE = level2_rmse ))
```

Level 3 - User Bias:

From our previous analysis 5.1 that many users tend to rate high all movies. We can take as example the mean rates higher than 3 which represents most users.

We can improve our model by including the user effect:

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

```
#first, estimate the effect of each movie bi
mu <- mean(edx$rating)

user_avgs <- edx %>%
  group_by(userId) %>%
  summarise(b_ii = mean(rating - mu))

# then, make the prediction using the new model y= mu + b
predicted_ratings <- mu + validation %>%
  left_join(user_avgs, by='userId') %>%
  pull(b_ii)

level3_rmse <- RMSE(predicted_ratings, validation$rating)
level3_rmse
```

```
## [1] 0.9785383
```

```
# Add the rmse result of the new model to the rmse data frame
rmse_conclusions <- bind_rows(rmse_conclusions,
                              data_frame(method="Level3 - User Bias:",
                                         RMSE = level3_rmse ))
```

Level 4 - Movie & User Bias:

We can improve our model by impacting the movie and user effect together:

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

```r
# Let's compute the new term bu in our mode y= mu+bi+bu +eps
mu <- mean(edx$rating)

movie_user_mean  <- edx %>%
  left_join(movie_avgs,n, by='movieId') %>%
  group_by(userId) %>%
  summarise(b_u = mean(rating - mu - b_i))

# Now, compute the predicted ratings using the new model with user and movie effects:
predicted_ratings <- validation %>%
  left_join(movie_avgs, by='movieId') %>%
  left_join(movie_user_mean, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)

level4_rmse <- RMSE(predicted_ratings, validation$rating)
level4_rmse
```

```
## [1] 0.8656398
```

```r
# Add the rmse result of the new model to the rmse data frame
rmse_conclusions <- bind_rows(rmse_conclusions,
                              data_frame(method="Level4 - User & Movie Bias:",
                                         RMSE = level4_rmse ))
```

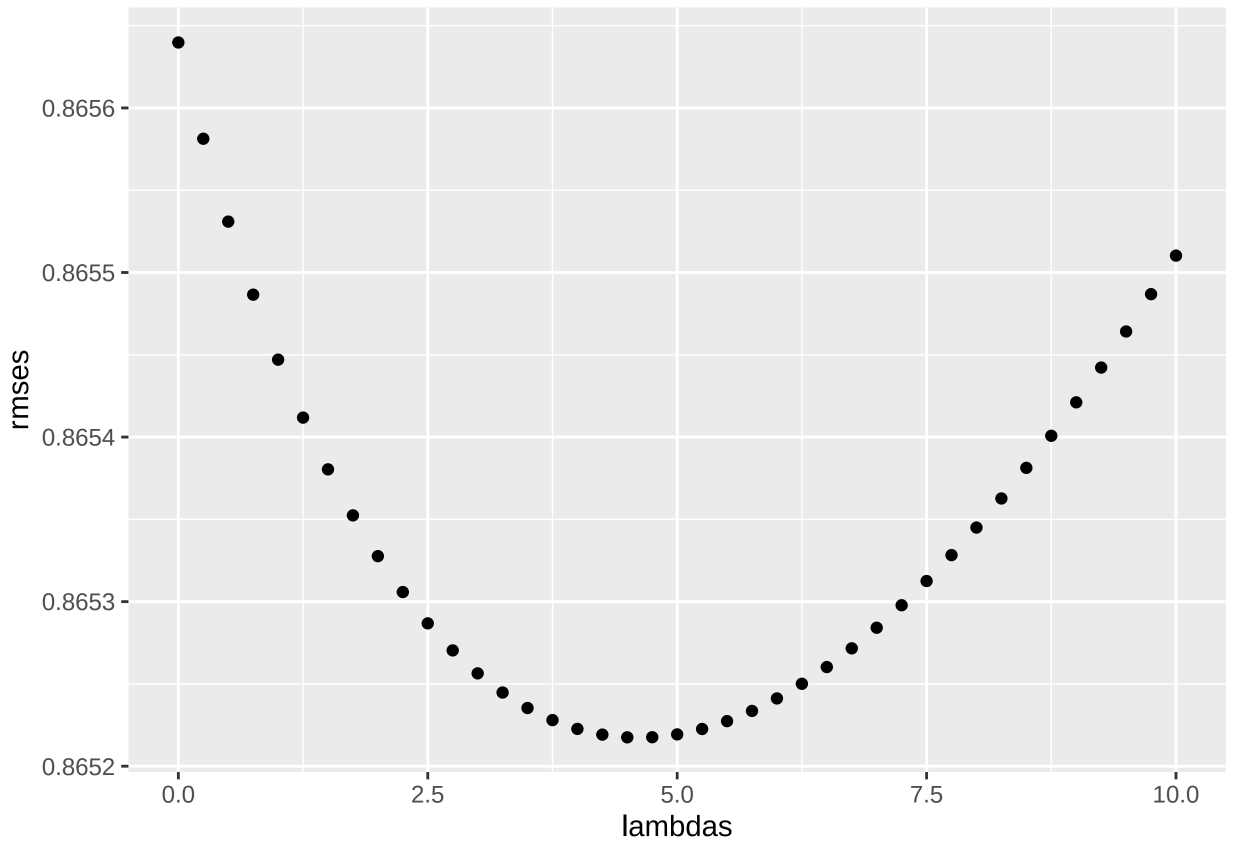
```r
rmse_conclusions
```

```
## # A tibble: 4 x 2
##   method                       RMSE
##   <chr>                       <dbl>
## 1 Level1 - Mean:              1.06
## 2 Level2 - Movie Bias:        0.944
## 3 Level3 - User Bias:         0.979
## 4 Level4 - User & Movie Bias: 0.866
```

##Level 5 - Regularization:

```r
lambdas <- seq(0, 10, 0.25)

rmses <- sapply(lambdas, function(l){
  mu <- mean(edx$rating)

  b_i <- edx %>%
    group_by(movieId) %>%
    summarize(b_i = sum(rating - mu)/(n()+l))

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

  predicted_ratings <-
```

```
    validation %>%
    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, validation$rating))
})

qplot(lambdas, rmses)
```



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

```
## [1] 4.5
```

```
level5_rmse <- min(rmses)

rmse_conclusions <- bind_rows(rmse_conclusions,
                              data_frame(method="Level5 - Regularization:",
                                         RMSE = level5_rmse ))
rmse_conclusions
```

```
## # A tibble: 5 x 2
##   method                       RMSE
##   <chr>                        <dbl>
## 1 Level1 - Mean:               1.06
## 2 Level2 - Movie Bias:         0.944
## 3 Level3 - User Bias:          0.979
## 4 Level4 - User & Movie Bias: 0.866
## 5 Level5 - Regularization:    0.865
```

Conclusion

The capstone project developed in the Data Science Certificate (PH125.9x) presented in this report was to create a movie recommendation system. The data used in this project is the the MovieLens 10M dataset. Data was downloaded, processed, cleaned, analyzed and stored. The recommendation system was based on RMSE function that was optimized by user and movie effects and by regularisation process.