MovieLens

Ricardo Zuñiga

2/9/2019

Executive summary

Welcome to this project, I hope be interesting for all readers, the main goal is create a recomendation system that would predict the rating of some user would give to a certain movie based on the movie itself, the date, the genre and user preferences.

The principal Data Set for the project is "MovieLens",it contain 10Million of observations in where includes , which user to a specific movie gives certain rate(from 0 to 5 stars), where we gone split the data into a training set and test set.

Proposed way for measure the accuracy of algorithm is "RMSE" (root mean squared error), basically this form calculates difference between our prediction and the true result taking that difference is squared to obtain optimization, so in this project To be considered successful we need the result of the RMSE to be less than 0.8649.

The key steps will be: -Identify parameters which are more important when making predictions. -Get each variable bias. -Create a regularization function to prevent movies or users with very little observations from falling errors due to high standard deviation. -Re-design the Data Set so there is no more than one genre per observation. -Save the results to work with them. -Results different comparison. -Algorithm possible improvements

#Obtain Data Set

Know is time to download the Data set and create the Data partition, in this section will use the code provided by EDX.

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
library(data.table)
library(tidyverse)
library(dplyr)
library(caret)
library(lubridate)

# 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) #when data is downloaded
#ratings <- fread(text = gsub("::", "\t", readLines(unzip(dl, "ml-10M100K/ratings.dat"))),
#                 col.names = c("userId", "movieId", "rating", "timestamp"))
ratings <- fread(text = gsub("::", "\t", readLines("ml-10M100K/ratings.dat")),
                 col.names = c("userId", "movieId", "rating", "timestamp"))
```

```
#movies <- str_split_fixed(readLines(unzip(dl, "ml-10M100K/movies.dat")), "\\::", 3)
movies <- str_split_fixed(readLines("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")

# Validation set will be 10% of MovieLens data

#set.seed(1, sample.kind="Rounding")
# if using R 3.5 or earlier, use `
set.seed(1)
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]

# Make sure userId and movieId in validation set are also in edx set

validation <- temp %>%
  semi_join(edx, by = "movieId") %>%
  semi_join(edx, by = "userId")

# Add rows removed from validation set back into edx set

removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)

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

Once the code is loaded and Data Partition created its time to analize the data and define which variables are important, also how algorithm would be design, finishing analysis step would go to results discussion.

Involving with the Data

To better understand what information are going to work with and know more about its properties, it is necessary to do a small exploration and then move on to an analysis. First, a small exploration of the data set where we can see the number of users, the number of movies, what trend of ratings you have, if it is normal, or if it is variable.

First, the number of columns and rows that the EDX set has is analyzed.

```
ncol(edx)  #show number of columns
## [1] 6
nrow(edx)  #show number of rows
## [1] 9000061
```

Second step, number of users who have rated 3 stars and 0 stars to a movie is observed.

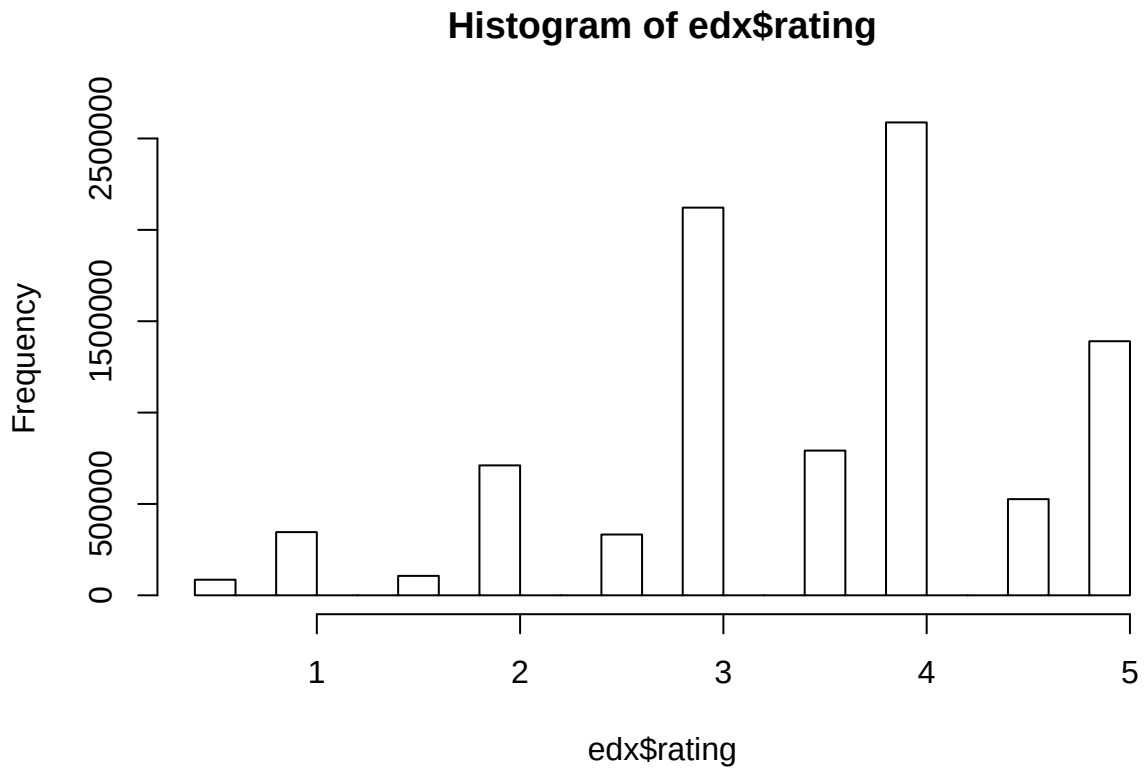
```
filter(edx,rating=="3")%>% summarize(n()) #number of 3 stars
##       n()
## 1 2121638
filter(edx,rating=="0")%>% summarize(n()) #number of 0 stars
##   n()
## 1   0
```

Third step, see how many different users and movies EDX set has

```
x<-data.frame(unique(edx$movieId))
nrow(x)
## [1] 10677
y<-data.frame(unique(edx$userId))
nrow(y)
## [1] 69878
```

Finally, to realize if we are working with a normal rates distribution, should see the amount of films that exist in several genres as well as generate a score histogram.

```
hist(edx$rating)
```



Analysis

Then will analyze each variable bias, in this case will be movie, user, genre and view date, however, to create a bias for date and genre have certain problems where will have to clean the code a bit, both in train set and in test set.

Data cleaning consists in first separating all existing genres in an observation to new observations, it means, create a new observation for each genre reported in the movie Id of a row, for example suppose we have an observation with movie Id "z"And in genres column there is the notation x1 | x2 | x3, starting this to create the observation z, x1, the observation z, x2 and the observation z, x3.

Now be able to work with the date column it is somewhat simpler since now only is need to specify the time column of the observations come in a time class variable and round units to the nearest week.

```
edx_sep_by_genre<-edx %>% separate_rows(genres,sep="\\|")
edx_sep_by_genre$timestamp<-as_datetime(edx_sep_by_genre$timestamp)
```

With edx data set clean next step is to analyze each variable bias, this is done with function "left_join" of the created data and grouping them by the new variable to be used, the average is calculated and the bias from previous variables are subtracted.

```
mu <- mean(edx$rating)

movie_edx <- edx%>%
  group_by(movieId) %>%
  summarize(b_i = mean(rating - mu))
  b_i_mean<-mean(movie_edx$b_i) #calculate the mean bias per movie

#calculate the bias by user
user_edx <- edx %>%
  left_join(movie_edx, by='movieId') %>%
  group_by(userId) %>%
  summarize(b_u = mean(rating - mu - b_i))
  b_u_mean<-mean(user_edx$b_u) #calculate the mean bias per user



#calculate the bias by genre
 genre_edx <- edx_sep_by_genre %>%
  left_join(movie_edx, by='movieId') %>%
  left_join(user_edx, by='userId') %>%
  group_by(genres)%>%
  summarize(va = mean(rating - mu - b_i-b_u))

#calculate the bias by date
date_edx<-edx_sep_by_genre %>%
  left_join(movie_edx, by='movieId') %>%
  left_join(user_edx, by='userId') %>%
  left_join(genre_edx,by='genres')%>%
  mutate(date = round_date(timestamp, unit = "week")) %>%
  group_by(date) %>%
  summarize(b_d = mean(rating- mu - b_i- b_u- va))
```

In this way is possible generate our firsts approximations that are adding the average, plus each of the bias, the RMSE functions applied against the real values, each result must be saved into a new data frame observation.

```
#separate genres from the validation set
validation_genres<-validation%>%separate_rows(genres,sep="\\|")
validation_genres$timestamp<-as_datetime(validation_genres$timestamp)



#calculate the RMSE
```

```
results<-data.frame() #create a data frame where results would be saved
# variable format as "results[x,1]" is the name of the observation case we are predicting
#variable format as "results[x,2]" is the RMSE result of the observation case

#case 1 just by mean

results[1,1]<-"mean"
predicted_ratings<-validation%>%mutate(pred=mu)%>%pull(pred)
results[1,2]<-RMSE(predicted_ratings,validation$rating)
#results[1,2]<-1
#case 2 just by mean and movie bias
results[2,1]<-"movie b"
predicted_ratings <- validation_genres %>%
  left_join(movie_edx, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
results[2,2]<- RMSE(predicted_ratings, validation_genres$rating)

#case 3 just by mean, movie and user bias
results[3,1]<-"user b"
predicted_ratings <- validation_genres %>%
  left_join(movie_edx, by='movieId') %>%
  left_join(user_edx, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
results[3,2]<- RMSE(predicted_ratings, validation_genres$rating)

# case 4 just by mean, movie, usear and genre bias
results[4,1]<-"genre b"
predicted_ratings <- validation_genres %>%
  left_join(movie_edx, by='movieId') %>%
  left_join(user_edx, by='userId') %>%
  left_join(genre_edx,by='genres') %>%
  mutate(pred = mu + b_i + b_u+va) %>%
  pull(pred)
results[4,2]<- RMSE(predicted_ratings, validation_genres$rating)

# case5 whole bias (mean, movie, user, genre and date)
results[5,1]<-"date b"

predicted_ratings <- validation_genres %>%
left_join(movie_edx, by='movieId') %>%
left_join(user_edx, by='userId') %>%
left_join(genre_edx,by='genres') %>%
  mutate(date = round_date(timestamp, unit = "week")) %>%
  left_join(date_edx,by='date')%>%
  mutate(b_if=ifelse(is.na(b_i),b_i_mean,b_i))%>%          #It is not neccesary just in the case the al
  mutate(b_uf=ifelse(is.na(b_u),b_u_mean,b_u))%>%          #It is not neccesary just in the case the al
mutate(pred = mu + b_if + b_uf+va+b_d) %>%
pull(pred)
results[5,2]<- RMSE(predicted_ratings, validation_genres$rating)
```

There are already data and a rating predict algorithm of movies, based on what has been said above, however

exists movies or users that rarely appear in edx data set, so to create an attempt to pattern regularization for those movies Due to their large standard deviation they could affect the accuracy of the entire algorithm.

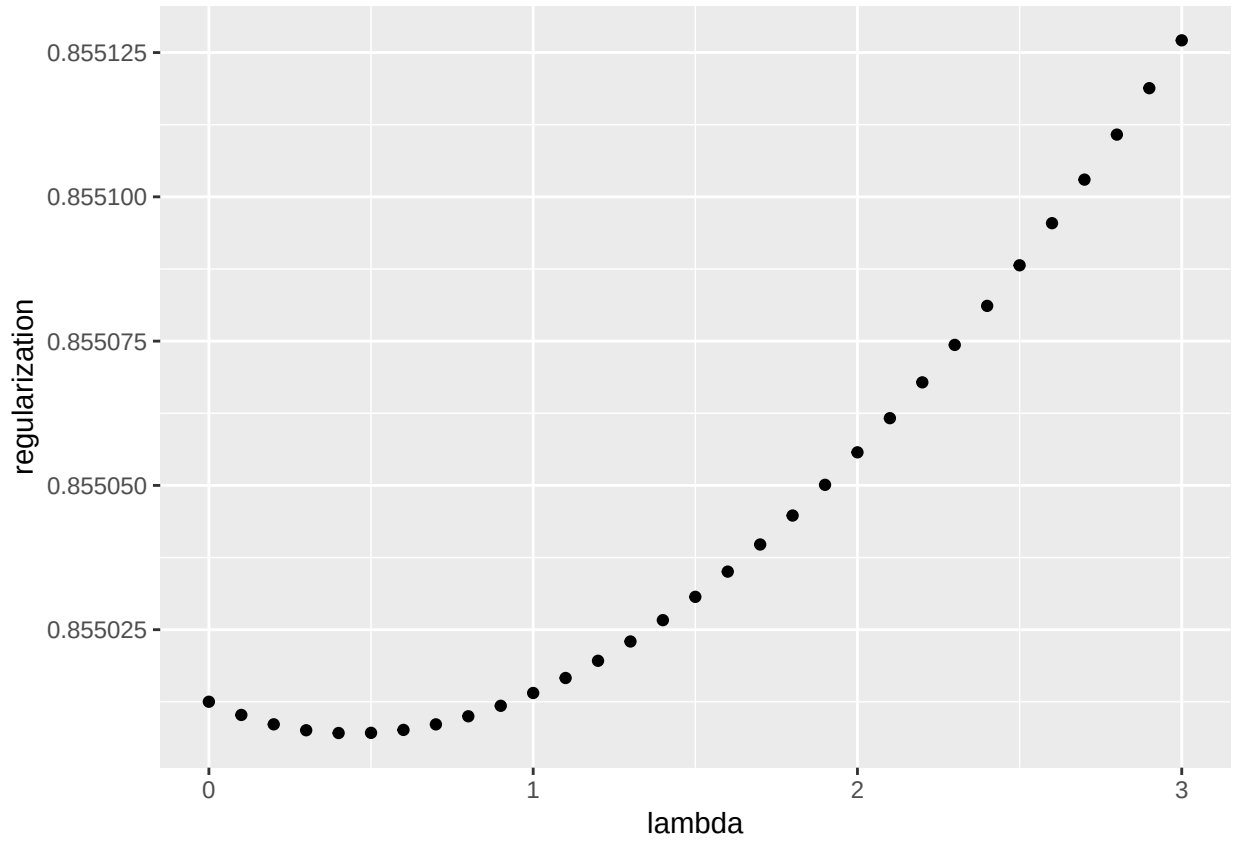
For create regularization we have to program a function where will assign a lambda sequence values used to calculate the elements sum between number of observations plus lambda, so find a value that optimizes the process and creates an RMSE must be minor, the function is:

```
lambda<-seq(0,3,0.1) #vector in which we gone to try fit the best lambda

regularization <- sapply(lambda, function(x){

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

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

  va<- edx_sep_by_genre %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    group_by(genres)%>%
    summarize(va = sum(rating - b_i - mu- b_u)/(n()+x))

  b_d<-edx_sep_by_genre %>%
    left_join(b_i, by='movieId') %>%
    left_join(b_u, by='userId') %>%
    left_join(va, by='genres')%>%
    mutate(date = round_date(timestamp, unit = "week")) %>%
    group_by(date) %>%
    summarize(b_d = sum(rating - b_i - mu-b_u-va)/(n()+x))

  predicted_ratings <-
    edx_sep_by_genre %>%
    left_join(b_i, by = "movieId") %>%
    left_join(b_u, by = "userId") %>%
    left_join(va, by = "genres") %>%
    mutate(date = round_date(timestamp, unit = "week")) %>%
    left_join(b_d, by = "date") %>%
    mutate(pred = mu + b_i + b_u+va+b_d) %>%
    pull(pred)

  return(RMSE(predicted_ratings, edx_sep_by_genre$rating))
})
```

Results

Now that function finds minimum regularization factor lambda, I plot all lambda values vs RMSE for assign to reg variable the lowest one.

```
qplot(lambda, regularization) #plot all lambdas to the RMSE provided
```




```
reg<-lambda[which.min(regularization)]  #choose the best lambda value
reg
## [1] 0.4
```

In plot you can see lambdas trend, at the beginning plot make a small parable and then increase linearly.

Now to get all the results previous code (each case) must be used and add the regularization value

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

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

genre_edx_reg<- edx_sep_by_genre %>%
  left_join(movie_edx_reg, by='movieId') %>%
  left_join(user_edx_reg, by='userId') %>%
  group_by(genres)%>%
  summarize(va = sum(rating - b_i - mu- b_u)/(n()+reg))
```

```
date_edx_reg<-edx_sep_by_genre %>%
  left_join(movie_edx_reg, by='movieId') %>%
  left_join(user_edx_reg, by='userId') %>%
  left_join(genre_edx_reg, by='genres')%>%
  mutate(date = round_date(timestamp, unit = "week")) %>%
  group_by(date) %>%
  summarize(b_d = sum(rating - b_i - mu-b_u-va)/(n()+reg))


#case 6 just by mean and movie bias
results[6,1]<-"movie reg"
predicted_ratings <- validation_genres %>%
  left_join(movie_edx_reg, by='movieId') %>%
  mutate(pred = mu + b_i) %>%
  pull(pred)
results[6,2]<- RMSE(predicted_ratings, validation_genres$rating)

#case 7 just by mean, movie and user bias
results[7,1]<-"user reg"
predicted_ratings <- validation_genres %>%
  left_join(movie_edx_reg, by='movieId') %>%
  left_join(user_edx_reg, by='userId') %>%
  mutate(pred = mu + b_i + b_u) %>%
  pull(pred)
results[7,2]<- RMSE(predicted_ratings, validation_genres$rating)

#case 8 just by mean, movie, usear and genre bias
results[8,1]<-"genre reg"
predicted_ratings <- validation_genres %>%
  left_join(movie_edx_reg, by='movieId') %>%
  left_join(user_edx_reg, by='userId') %>%
  left_join(genre_edx_reg,by='genres') %>%
  mutate(pred = mu + b_i + b_u+va) %>%
  pull(pred)
results[8,2]<- RMSE(predicted_ratings, validation_genres$rating)

#case 9 whole bias regulated
results[9,1]<- "Final"
predicted_ratings_reg <-
  validation_genres %>%
  left_join(movie_edx_reg, by = "movieId") %>%
  left_join(user_edx_reg, by = "userId") %>%
  left_join(genre_edx_reg, by = "genres") %>%
  mutate(date = round_date(timestamp, unit = "week")) %>%
  left_join(date_edx_reg, by = "date") %>%
  mutate(pred = mu + b_i + b_u+va+b_d) %>%
  pull(pred)

results[9,2]<- RMSE(predicted_ratings_reg, validation_genres$rating)
```

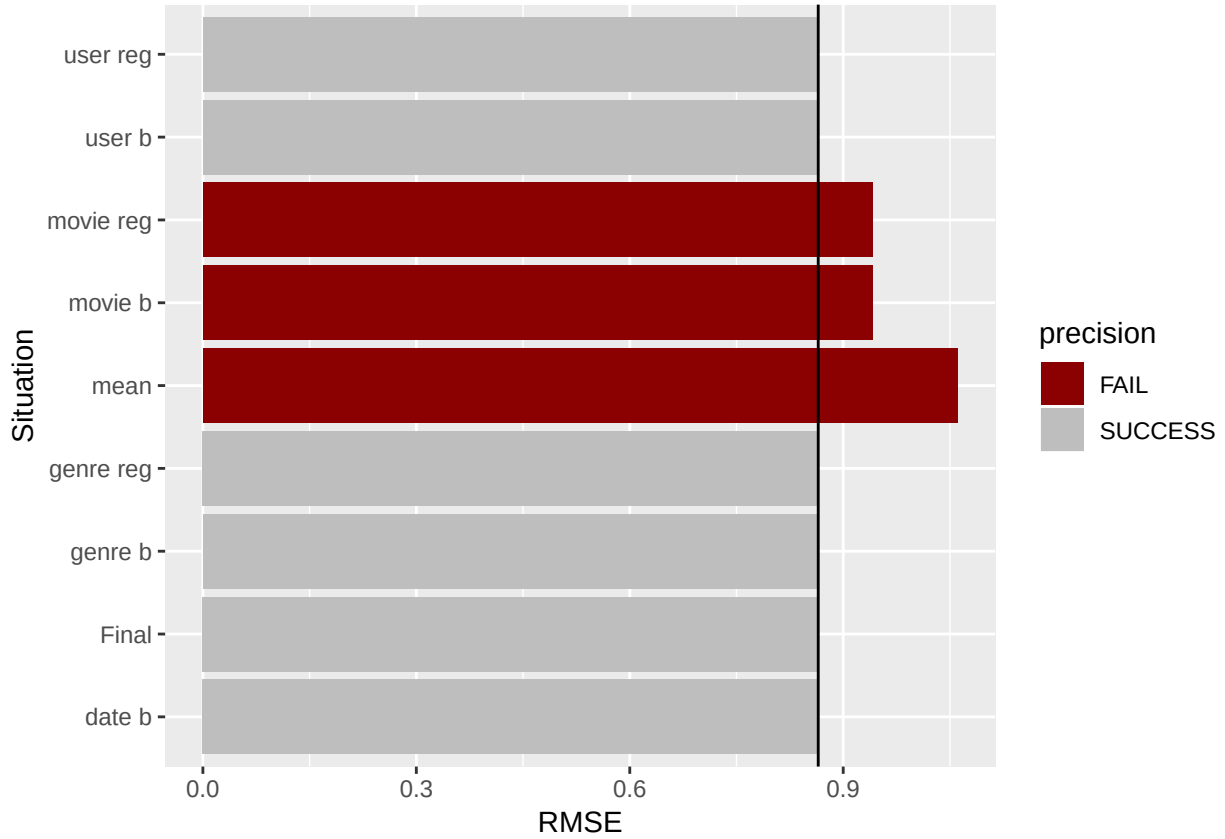
Next step is compare results, columns name is added to be able to organize them better, after print a table with the cases in descending order of RMSE.

```r
colnames(results)<-c("Situation","RMSE")

results<-results%>%arrange(desc(RMSE)) #sort cases down
results %>% knitr::kable() #all cases shown in a table
```

Situation	RMSE
mean	1.0606506
movie b	0.9411063
movie reg	0.9410933
user b	0.8634080
genre b	0.8633317
user reg	0.8633194
date b	0.8632593
genre reg	0.8632443
Final	0.8631701

The Final case (with the bias of all the variables, and regularization) is the smallest. To better observe the information obtained with the table, I will graph the values and automatically mark proposed RMSE exceed values to consider a successful experiment.

```r
results<-results%>%mutate(precision=ifelse(RMSE<0.8649,"SUCCESS","FAIL")) #categorize results as succes

ggplot(results,aes(Situation,RMSE))+
geom_col(aes(fill = precision))+
  geom_hline(yintercept = 0.8649,color='black',size = .5)+
  coord_flip()+
  scale_fill_manual(values = c("darkred","grey"))
```



In this way is easy to realize the algorithm improvement, cases that meet requirement are: -movie + user bias - user regularizated - movie+ user + genre bias - genre regularizated -movie + user + genre+ date bias - all bias + regularization (Final)

Project final RMSE corresponds to observation 9 of our table, where we have all bias included and we regularize it, the final RMSE is:

```
min(results$RMSE)
## [1] 0.8631701
```

Discussion and future work

Was proved in this project, it's possible to generate a algorithm film recommender algorithm based on ratings, with an RMSE less than 0.8649, was verified that as we added different bias with its regularization RMSE decrease, however, it arrived after 4 attempts the Algorithm reached a stagnation point in performance due to RMSE reduction was too small. I can compare this to the fact that the first iteration that met requirements resulted in "0.863408" and the final "0.863191", making us have an improvement of 0.025% along the other iterations.

Analyzing this little improvement percentage by adding more layers makes it known that there is future work for algorithm improvement, it means, we cannot continue with the same process carried out during the project, so it can be improved with another type of analysis. First option would be use caret package train function since it has a better precision however, we would occupy an equipment that can work with that amount of information efficiently, a second proposal is to use the "recommenderLab" packages to separate the information in an array larger and create correlations between several movies.

General Conclusion

films ratings are not a normally distributed variable, however, it is not a rating that is delivered so randomly, based on the evidence registered is possible to create a pattern about the user movies, release dates, seen time as well as the genre which gives us an idea about which person can like or dislike a certain film.

I keep in mind that the algorithm also has to be able to create a recommendation to a new user or movie over time, so exist the necessity to create a variable that helps us interpret the rating without just relying on the film or the average, in this work it was proposed to use the average of the bias of the film and keep the others, it will be in any way very imprecise, but it's a first approximation.

In summary, this work helps us to make predictions for near future with a certain degree of accuracy, so this experiment confirms that based on the previously data someone could create patterns to follow which tends to predict future in a right way.

Acknowledgments

I would like thank Professor Rafael A. Irizarry for guiding us and introducing us along path of Data science, in the same way special thanks to the entire Edx and HarvardX team for always being on the lookout and answering our questions, finally to all the classmates of the course we were together throughout this trip.

##References.

Rafael A. Irizarry.(2019). Introduction to Data Science. Data Analysis and Prediction Algorithms with R, HARVARDX, web site: https://rafalab.github.io/dsbook/index.html