title: "Capstone course"
author: "ubatifce"
date: "November 1 , 2019"

output: "html_document"

*** INDEX
*** 1. Introduction
*** 2.Executive Summary Section
*** 2.1 *Dataset:
*** 2.2 *****Goal of the project
*** 2.3 *Key steps that w ere performed
*** 3. Methods/analysis section
*** 3.1 *Process and techniques used,
*** 3.2 ******Data cleaning
*** 3.3 Data exploration
* **3.4*****Visualization and insights gained
*** 3.5 Modeling approach
*** 4. ***Results section ***
*** 4.1 *Modeling results and discusses
*** 4.2 *****Model performance
*** 5. Conclusion section
*** 5.1 *****Summary of the report
*** 5.2 *Limitations
*** 5.3 *****Future w ork

1.Introduction

Hello, fellow students.
My name is Marcello I am submitting my w ork for your evaluation.
I hope the w ork could be clear enough for you to understand it w ithout effort, at least, I'll do my best to get this objective.

As it is said by the staff,
**"The ability to clearly communicate the process and insights gained from an analysis is an important skill for data scientists. **

The project itself is based on the postulates that w ere given to us at the capstone module of the Data Science Professional Certificate.
My submission for this project is three files:
A report in PDF format: Report.pdf
A report in Rmd format: Report.Rmd
A script in R format: Script.R

We've been told to present a script in R format that generates the predicted movie ratings and RMSE score
This is an individual w ork. When I say "w e" I mean the students w ho are in this course. We are all doing the same project.

Files are expected to be dow nloaded from the uploaded files using the edx submision form.
As the staff recommend also providing a link to a GitHub repository containing the three files above, I also include links to a repository.

If one or more of these files are damaged or not available,an alternative w ay to dow nload them is:
All the files are in the public access repository: https://github.com/ubatifce/capstone

```
https://raw.githubusercontent.com/ubatifce/capstone/Report.pdf
https://raw.githubusercontent.com/ubatifce/capstone/Report.Rmd
https://raw.githubusercontent.com/ubatifce/capstone/Script.R
or
https://github.com/ubatifce/capstone/blob/master/Report.Rmd
https://github.com/ubatifce/capstone/blob/master/Report.pdf
https://github.com/ubatifce/capstone/blob/master/Script.R
```

Clone w ith HTTPS: Use Git or checkout w ith SVN using the w eb URL. https://github.com/ubatifce/capstone.git

#***2. Executive Summary Section ***

*Dataset:
A set of datasets w ere given to us by the staff members.
The ones that w ere used w ere the standard ones, all of the based on the MovieLens database, and dow nloaded in the ml-10M100K.zip file.
Anyw ay, in my script, I used the automatic dow nloading procedure so I got:

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
temp <- movielens[test_index,]
validation <- temp %>%
     semi_join(edx, by = "movieId") %>%
     semi_join(edx, by = "userId")
```

*Goal of the project
As it w as already said, the students have to w rite a script in R format that generates the predicted movie ratings and RMSE score
The goal of the project w ill be then to create a report and a script that has a proper performance calculating the RMSE of the dataset

*Key steps that w ere performed
Instructions say that most learners choose to display all code chunks in the report, but it is not required.
We only have to be sure that the report includes the RMSE
I take this into account.
The first step performed w as to create a sub set from movielens, called edx
The second step performed w as to create a sub set for validations from movielens, called validation
The third step w as to make sure that validation has the same movieIDs and userIds than edx

#***3. Methods/analysis section ***
*Process and techniques used,
Instructions say : “Develop your algorithm using the edx set. For a final test of your algorithm, predict movie ratings in the validation set ”
I follow these instructions.

*Data cleaning
The first data cleaning uses edx dataset, from movielens, as:
#given movielens <- left_join(ratings, movies, by = "movieId") dow nloaded from the initial script

```
test_index <- createDataPartition(y = movielens$rating, times = 1, p = 0.1, list = FALSE)
edx <- movielens[-test_index,]
removed <- anti_join(temp, validation)
edx <- rbind(edx, removed)
```

It also uses validation, as

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

*Data exploration

A generic (taken from theory) RMSE function looks like this.

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

*Visualization and insights gained

In order to see the behaviour of the RMSE tow ards the lambdas, the script contain orders like:
**** Relationship Lambda vs RMSE

```
 qplot(lambdas, rmses)
```

**** Relationship Originals vs Regularized

```
 data_frame(Original = movie_avgs$b_i,
           Regularized = movie_reg_avgs$b_i,
           n = movie_reg_avgs$n_i) %>%
     ggplot(aes(Original, Regularized, size=sqrt(n))) +
     geom_point(shape=3, alpha=0.25, color="red")
```

*Modeling approach

As the instructions say, I had to be sure not to use the validation set for training or regularization.
It is also noted that it is said that "you may w ish to create an additional partition of training and test sets from the provided edx dataset to experiment w ith multiple parameters… "
This w ill be taken into account.

I used this approach to experiment w ithout using validation set for training or regularization.
I created smaller train and test sets, w ith these characteristics:

****training and test sets names
In order not to use validation set for training, I created to experiment:

```
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(y = edx$rating, times = 1,
                                  p = 0.1, list = FALSE)
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
```

****baseline predictors and regularization
I used mu as the mean of the train set ratings ; mu <- mean(train_set$rating)
I used lambda as a sequence from 0 to 10 at intervals of 0.125 lambda<- seq(0, 10, 0.125)
lambdas
[1] 0.000 0.125 0.250 0.375 0.500 0.625 0.750 0.875 1.000 1.125
[11] 1.250 1.375 1.500 1.625 1.750 1.875 2.000 2.125 2.250 2.375

[21] 2.500 2.625 2.750 2.875 3.000 3.125 3.250 3.375 3.500 3.625
[31] 3.750 3.875 4.000 4.125 4.250 4.375 4.500 4.625 4.750 4.875
[41] 5.000 5.125 5.250 5.375 5.500 5.625 5.750 5.875 6.000 6.125
[51] 6.250 6.375 6.500 6.625 6.750 6.875 7.000 7.125 7.250 7.375
[61] 7.500 7.625 7.750 7.875 8.000 8.125 8.250 8.375 8.500 8.625
[71] 8.750 8.875 9.000 9.125 9.250 9.375 9.500 9.625 9.750 9.875
[81] 10.000

****models used

1.Using AVG of (train ratings)
2 Using AVG(rating- AVG(train ratings))
3 Using AVG(rating-AVG(train) ratings)-mean(rating - mu)
4 Using sum(rating - mu)/(n()+lambda), n_i = n()
5 Using sum(rating - mu)/(n()+lambda), n_i = n() with min(lambda)

#Best shots and RMSEs

method **RMSE

n=1000
Using set.seed(1, sample.kind="Rounding") Using Method: sum(rating - mu)/(n()+lambda), n_i = n() with min(lambda) 1.09868
Using seed(1) Using Method:sum(rating - mu)/(n()+lambda), n_i = n() with min(lambda) 1.098680 ****No change with previous one
Using seed(755) Using Method: AVG(rating-AVG(train) ratings)-mean(rating - mu) 0.8994061 ****seed affects RMSE!!!!

All other test are Using set.seed(1, sample.kind="Rounding")
n=10000
Using sum(rating - mu)/(n()+lambda), n_i = n() with min(lambda) [1] 0.8697629

n=400000
Using sum(rating - mu)/(n()+lambda), n_i = n() with min(lambda) | 0.8627683|

n=1000000
|Using sum(rating - mu)/(n()+lambda), n_i = n() with min(lambda) | 0.8736320|

n=5000000
|Using sum(rating - mu)/(n()+lambda), n_i = n() with min(lambda) | 0.8646980|

4. ***Results section ***

* 4.1 Modeling results and discusses

The final RMSE was 0.8642491
The model chosen was |Using sum(rating - mu)/(n()+lambda), n_i = n() with min(lambda)
After trainnig, lambda was able to be optimized.
At the bottom of this report, the algorithm is exposed.

4.2 *Model performance

The model was not changed by the use of seedsdid Using set.seed(1, sample.kind="Rounding") versus Using seed(1)
However, using other seeds as seed(755) shower a change in the output.
In this case, the method: AVG(rating-AVG(train) ratings)-mean(rating - mu) seemd to be the best alternative.
Finally, set.seed(1, sample.kind="Rounding") was chosen as standard seed for the project.
Making an analysis of n performance, we can see that after n=1x10^6 we get some stability of the model near 0.86/0.87

From n=5x10^6 mill to the final dataset, w e just get improvements in the third digit or low er.

5. Conclusion section

5.1 *Summary of the report

The goal of the project , in the sense of calculating a RMSE and print the valeus w as reached.
As it w as pointed earlier, The final RMSE calculated w as 0.8642491
The model chosen w as Using sum(rating - mu)/(n()+lambda), n_i = n() w ith min(lambda)

5.2 *Limitations

These models are very difficult to w ork w hen you are a student and you do not have access to the ultimates PC.
Using PCs w ith low memory ans low processors is possible, but consumes a lot of time.

5.3 *Future work

This is a nice subject to w ork w ith.
It w ould be temting to go on investigating the subjects once the course is finished.

Algorithm

RMSE algorithm. Function used:

**SEE POST DATA (at the end of the report) for the preliminary code that loads edx and validation:

In this project this RMSE function w ill be called using these arguments.

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

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

*RMSE FINAL MAIN algorithm.

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

# FINAL ALGORITHM

#training datasets
#test_index <- createDataPartition(y = edx$rating, times = 1,
                                  p = 0.1, list = FALSE)
#train_set <- edx[-test_index,]
#test_set <- edx[test_index,]
#test_set <- test_set %>%
#     semi_join(train_set, by = "movieId") %>%
#     semi_join(train_set, by = "userId")
```

```
# final datasets
train_set <- edx
test_set <- validation



#A generic (taken fron theory) RMSE function

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

#Methods used:



#5 Using sum(rating - mu)/(n()+lambda), n_i = n() with min(lambda)


mu <- mean(train_set$rating)
movie_avgs <- train_set %>%
     group_by(movieId) %>%
     summarize(b_i = mean(rating - mu))

user_avgs <- test_set %>%
     left_join(movie_avgs, by='movieId') %>%
     group_by(userId) %>%
     summarize(b_u = mean(rating - mu - b_i))

predicted_ratings <- test_set %>%
     left_join(movie_avgs, by='movieId') %>%
     left_join(user_avgs, by='userId') %>%
     mutate(pred = mu + b_i + b_u) %>%
     .$pred


test_set %>%
     left_join(movie_avgs, by='movieId') %>%
     mutate(residual = rating - (mu + b_i)) %>%
     arrange(desc(abs(residual))) %>%
     select(title,  residual) %>% slice(1:10)

movie_titles <- movielens %>%
     select(movieId, title) %>%
     distinct()

movie_avgs %>% left_join(movie_titles, by="movieId") %>%
     arrange(b_i) %>%
     select(title, b_i) %>%
     slice(1:10)


train_set %>% dplyr::count(movieId) %>%
     left_join(movie_avgs) %>%
     left_join(movie_titles, by="movieId") %>%
     arrange(b_i) %>%
     select(title, b_i, n) %>%
     slice(1:10)


mu <- mean(train_set$rating)
movie_reg_avgs <- train_set %>%
     group_by(movieId) %>%
     summarize(b_i = sum(rating - mu)/(n()+lambda), n_i = n())

data_frame(Original = movie_avgs$b_i,
           Regularized = movie_reg_avgs$b_i,
           n = movie_reg_avgs$n_i) %>%
     ggplot(aes(Original, Regularized, size=sqrt(n))) +
     geom_point(shape=3, alpha=0.25, color="red")


train_set %>%
```

```
    dplyr::count(movieId) %>%
    left_join(movie_reg_avgs) %>%
    left_join(movie_titles, by="movieId") %>%
    arrange(b_i) %>%
    select(title, b_i, n) %>%
    slice(1:10)


#5 Using sum(rating - mu)/(n()+lambda), n_i = n() with min(lambda)


lambdas <- seq(2.3, 2.5, 0.125)

rmses <- sapply(lambdas, function(l){
    mu <- mean(train_set$rating)
    b_i <- train_set %>%
        group_by(movieId) %>%
        summarize(b_i = sum(rating - mu)/(n()+l))
    b_u <- train_set %>%
        left_join(b_i, by="movieId") %>%
        group_by(userId) %>%
        summarize(b_u = sum(rating - b_i - mu)/(n()+l))
    predicted_ratings <-
        test_set %>%
        left_join(b_i, by = "movieId") %>%
        left_join(b_u, by = "userId") %>%
        mutate(pred = mu + b_i + b_u) %>%
        .$pred
    return(RMSE(predicted_ratings, test_set$rating))

})

qplot(lambdas, rmses)

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


rmse_results <- data_frame(method="Using sum(rating - mu)/(n()+lambda), n_i = n() with min(lambda)",
                                     RMSE = min(rmses)) #)
rmse_results %>% knitr::kable()
min(rmse_results[2])


# END FINAL ALGORITHM
```