ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ

ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

MSc in Business Analytics

Big Data Systems and Architectures

Redis & MongoDB Assignment

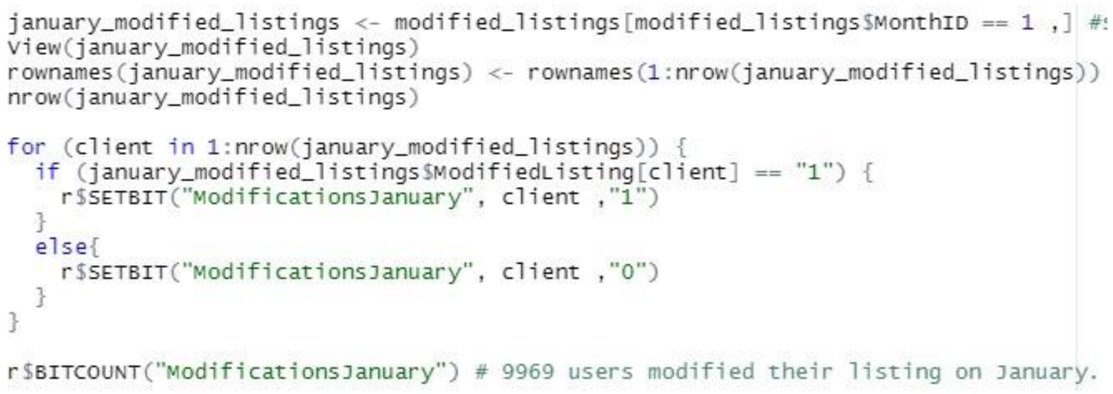
Name: Taklakoglou Chidiroglou Argyrios

Student ID: f 2822114

Redis

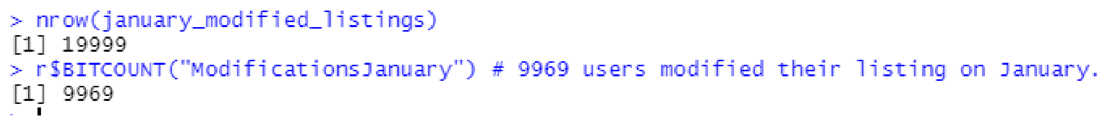
Task 1.1

In order to find out how many users have modified their list in January, I need to count how many UserIDs in dataframe modified_listings have the number 1 in the MonthID and ModifiedListing columns at the same time. Initially, from the modified_listings dataframe, I create a new dataframe that contains only the entries for January and then I iteratively check again in this new dataframe (january_modified_listings) for each userID if it has made a modification. That is, if the ModifiedListing column has the number 1. So, I create a Bitmap that contains 1 for each userID that has proceeded in a modification in January and 0 for each UserID that hasn't.



```
january_modified_listings <- modified_listings[modified_listings$MonthID == 1 ,] #
View(january_modified_listings)
rownames(january_modified_listings) <- rownames(1:nrow(january_modified_listings))
nrow(january_modified_listings)

for (client in 1:nrow(january_modified_listings)) {
  if (january_modified_listings$ModifiedListing[client] == "1") {
    r$SETBIT("ModificationsJanuary", client ,"1")
  }
  else{
    r$SETBIT("ModificationsJanuary", client ,"0")
  }
}

r$BITCOUNT("ModificationsJanuary") # 9969 users modified their listing on January.
```

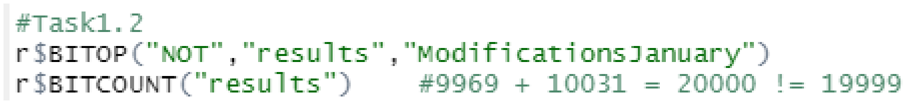
The result is that 9969 out of 19999 made a modification. Or otherwise almost 50% of userID.



```
> nrow(january_modified_listings)
[1] 19999
> r$BITCOUNT("ModificationsJanuary") # 9969 users modified their listing on January.
[1] 9969
```

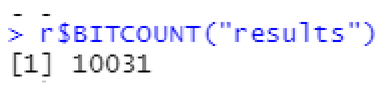
Task 1.2

To find out how many users did not make a change in their list in January, I only have to find the reverse of question 1.1. So, all I have to do invert the bitmap (ModificationsJanuary) of task 1.1. So, I convert 0 to 1 and vice versa.



```
#Task1.2
r$BITOP("NOT","results","ModificationsJanuary")
r$BITCOUNT("results")    #9969 + 10031 = 20000 != 19999
```

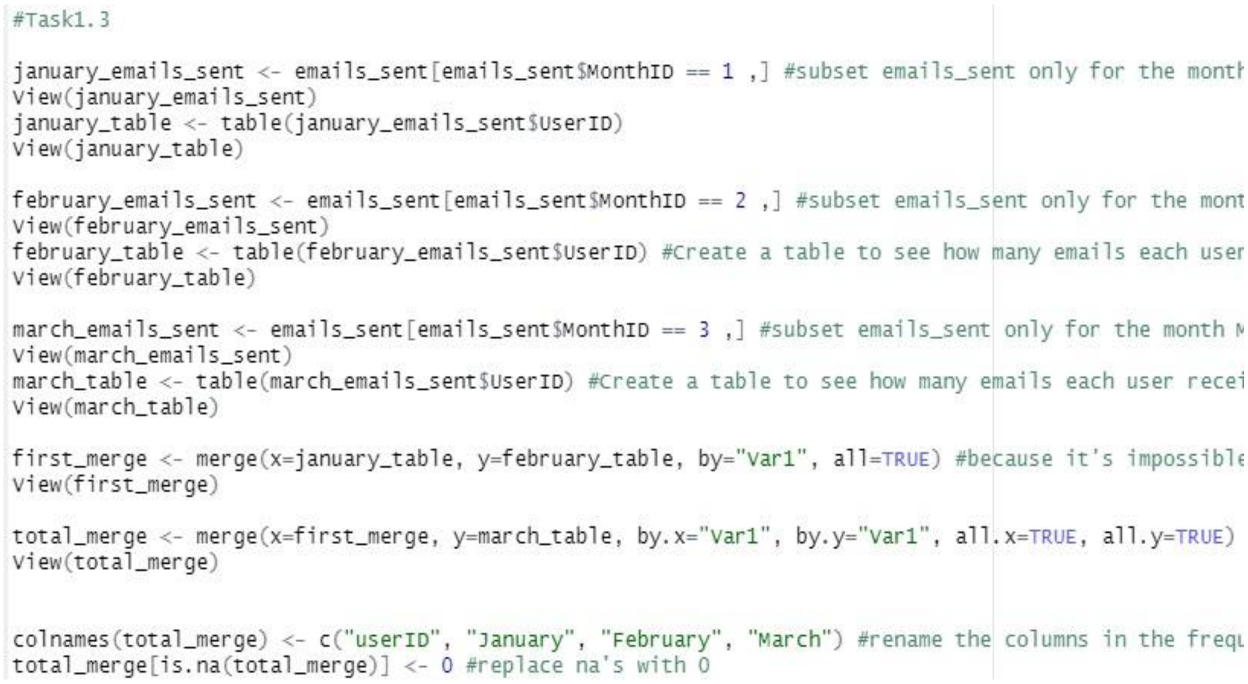
I notice that the result is 10031. However, the sum of the result of 1.1 and the result of 1.2 should be equal to 19999 and not equal to 20000. This happend because the results of Bitop must be a multiple of 8 as Bitop distributes the results in byte-levels where 1 byte is equal to 8 bits. So, in the bitmap of 1.1 another zero was created so that the total length (ace and zeros) is equal to 20000 which is a multiple of 8, but with the denial in question 1.2 the additional 0 became 1 and so instead of 10030 we received as answer 10031.



```
> r$BITCOUNT("results")
[1] 10031
```

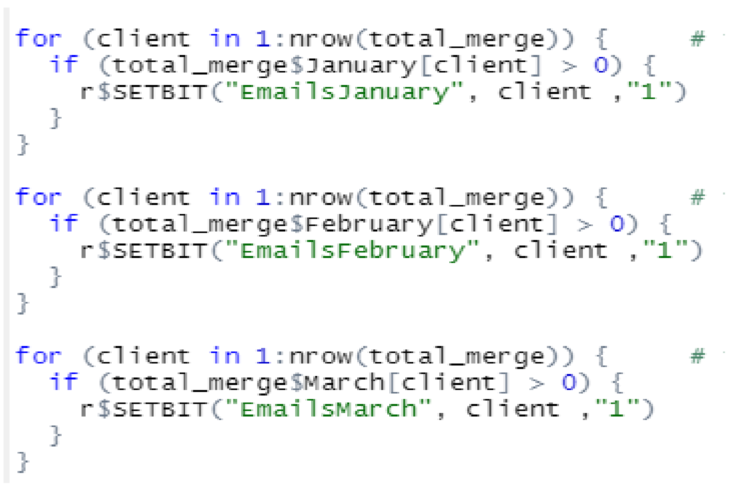
Task 1.3

In this task I want to find out how many userID received at least one email in January, in February and in March. Thereby, I create 3 new dataframes from dataframe emails_sent dataframe, where each one contains the userIDs who received email in this specific month. However, I notice that some userIDs have received an email more than one time in a month. Thus, I use the table command so as to observe how many emails each userID received each month (it will have received at least 1 as the userIDs that did not receive email in this month, will not be in the dataframe to which I applied the table command). Afterwards, in order to have a full knowledge about the total emails each userID received in the 3 months I am interested in, I will merge by using the merge command the 3 tables that have emerged from the table command. Because it is not possible to join all 3 of these tables directly, I first join the first two and then the third. I notice that the months when a userID didn't receive email, NA value appears. So, the next step is to replace the NAs with 0 and rename the columns.



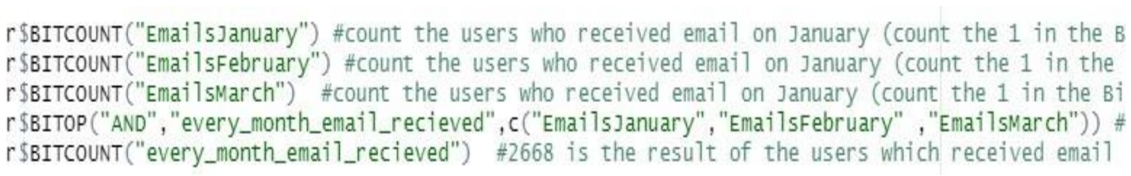
```
#Task1.3

january_emails_sent <- emails_sent[emails_sent$MonthID == 1 ,] #subset emails_sent only for the month
View(january_emails_sent)
january_table <- table(january_emails_sent$UserID)
View(january_table)

february_emails_sent <- emails_sent[emails_sent$MonthID == 2 ,] #subset emails_sent only for the mont
View(february_emails_sent)
february_table <- table(february_emails_sent$UserID) #Create a table to see how many emails each user
View(february_table)

march_emails_sent <- emails_sent[emails_sent$MonthID == 3 ,] #subset emails_sent only for the month M
View(march_emails_sent)
march_table <- table(march_emails_sent$UserID) #Create a table to see how many emails each user recei
View(march_table)

first_merge <- merge(x=january_table, y=february_table, by="Var1", all=TRUE) #because it's impossible
View(first_merge)

total_merge <- merge(x=first_merge, y=march_table, by.x="Var1", by.y="Var1", all.x=TRUE, all.y=TRUE)
View(total_merge)


colnames(total_merge) <- c("userID", "January", "February", "March") #rename the columns in the frequ
total_merge[is.na(total_merge)] <- 0 #replace na's with 0
```

Then I create 3 Bitmaps (one for each month) which contain 1 for the userIDs who received at least one email in this specific month and 0 if they did not receive an email.



```
for (client in 1:nrow(total_merge)) {      #
  if (total_merge$January[client] > 0) {
    r$SETBIT("EmailsJanuary", client ,"1")
  }
}

for (client in 1:nrow(total_merge)) {      #
  if (total_merge$February[client] > 0) {
    r$SETBIT("EmailsFebruary", client ,"1")
  }
}

for (client in 1:nrow(total_merge)) {      #
  if (total_merge$March[client] > 0) {
    r$SETBIT("EmailsMarch", client ,"1")
  }
}
```

Because I want to count the userIDs that received at least one email in January, in February and in March, I will use the Bitop AND command.

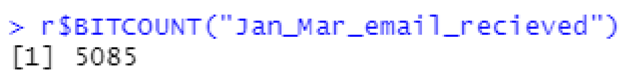


```
r$BITCOUNT("EmailsJanuary") #count the users who received email on January (count the 1 in the B
r$BITCOUNT("EmailsFebruary") #count the users who received email on January (count the 1 in the
r$BITCOUNT("EmailsMarch")  #count the users who received email on January (count the 1 in the Bi
r$BITOP("AND","every_month_email_recieved",c("EmailsJanuary","EmailsFebruary" ,"EmailsMarch")) #
r$BITCOUNT("every_month_email_recieved")  #2668 is the result of the users which received email
```

So, I conclude that 2668 are the userIDs who received at least one email in January, in February and in March.

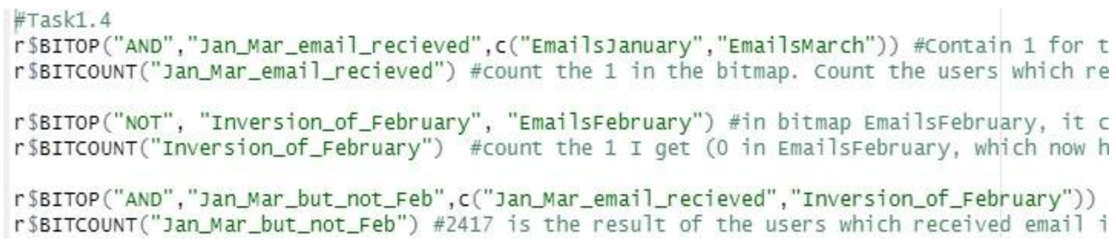
Task 1.4

In this task I want to find out how many userID received at least one email in January and in March but not in February. So, I will execute a Bitop AND command for the bitmaps corresponding to months January and March in order to have the common 1 of these two Bitmaps. I notice that 5085 userID have received emails in January and in March.



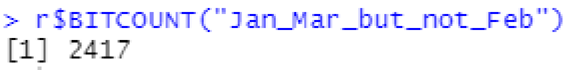
```
> r$BITCOUNT("Jan_Mar_email_recieved")
[1] 5085
```

Then I execute a Bitop NOT command for the month of February in order to convert the 0 to 1 and vice versa in the Bitmap of February. Finally, I execute a Bitop AND command for the Bitmaps of January and March and the reverse Bitmap for the month of February.



```
#Task1.4
r$BITOP("AND","Jan_Mar_email_recieved",c("EmailsJanuary","EmailsMarch")) #Contain 1 for t
r$BITCOUNT("Jan_Mar_email_recieved") #count the 1 in the bitmap. Count the users which re

r$BITOP("NOT", "Inversion_of_February", "EmailsFebruary") #in bitmap EmailsFebruary, it c
r$BITCOUNT("Inversion_of_February")  #count the 1 I get (0 in EmailsFebruary, which now h

r$BITOP("AND","Jan_Mar_but_not_Feb",c("Jan_Mar_email_recieved","Inversion_of_February"))
r$BITCOUNT("Jan_Mar_but_not_Feb") #2417 is the result of the users which received email i
```

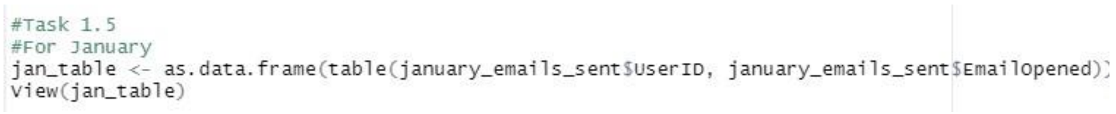
Thus, I conclude that 2417 userID received emails in January, March but not in February.



```
> r$BITCOUNT("Jan_Mar_but_not_Feb")
[1] 2417
```

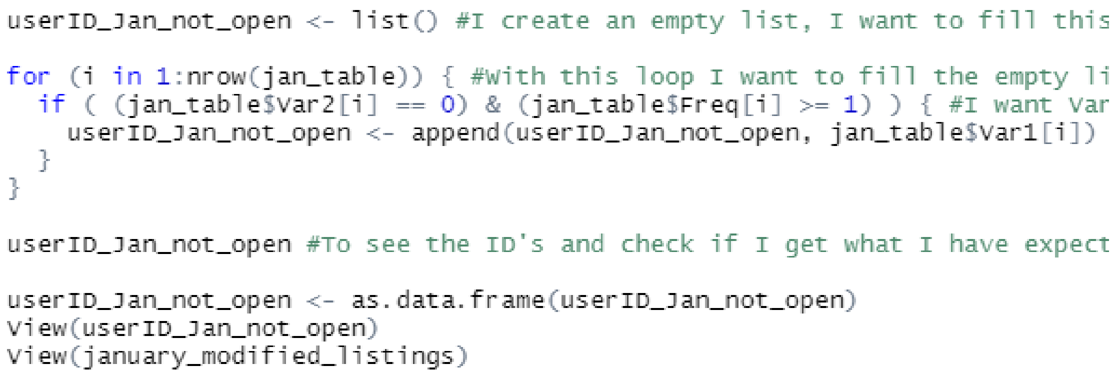
Task 1.5

The analysis that should be done in this question concerns the customers who received an email in January, did not open it but nevertheless made a modification in their list. Initially, by using the table variable for the userID and EmailOpened columns from the dataframe containing the userIDs that received emails in January, I create a dataframe with 3 columns. The first column refers to userIDs who received an email in January. The second column refers to whether each user has opened the email or not (1 if he has opened it and 0 if he has not opened it). The third column counts how many times this happens.
This way, I will know how many times each user who received an email in January opened it and how many did not. So, I have gathered all the information for each user that received an email in January.



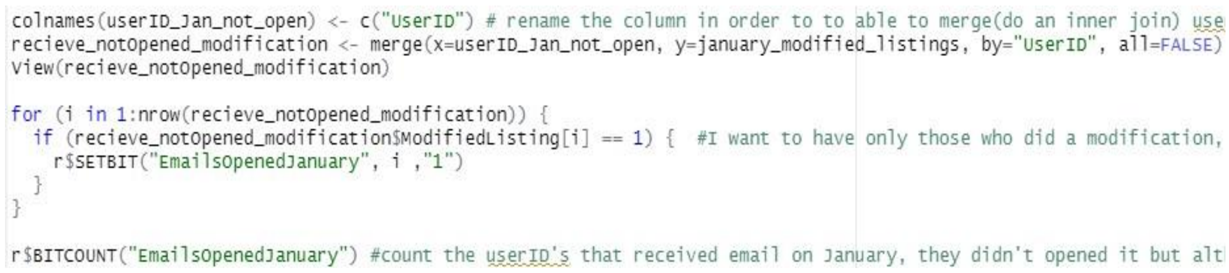
```
#Task 1.5
#For January
jan_table <- as.data.frame(table(january_emails_sent$UserID, january_emails_sent$EmailOpened))
View(jan_table)
```

Then I create a new dataframe (userID_Jan_not_open) that contains the userIDs who received the email in this specific month but did not open it..



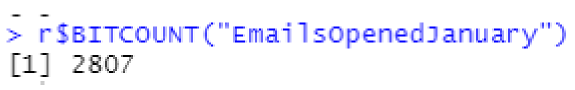
```
userID_Jan_not_open <- list() #I create an empty list, I want to fill this

for (i in 1:nrow(jan_table)) { #With this loop I want to fill the empty li
  if ( (jan_table$Var2[i] == 0) & (jan_table$Freq[i] >= 1) ) { #I want Var
    userID_Jan_not_open <- append(userID_Jan_not_open, jan_table$Var1[i])
  }
}

userID_Jan_not_open #To see the ID's and check if I get what I have expect

userID_Jan_not_open <- as.data.frame(userID_Jan_not_open)
View(userID_Jan_not_open)
View(january_modified_listings)
```

I want from dataframe january_modified_listings dataframe, the lines that indicate the users (userID) who did not open the email (s) they received in January. I do this by using the merge (inner) command for these two dataframes on the userID column. Finally I create a Bitmap with 1 for userIDs that have 1 in the ModifiedListing column of the new dataframe and 0 if they have 0 in the ModifiedListing column.



```
colnames(userID_Jan_not_open) <- c("UserID") # rename the column in order to to able to merge(do an inner join) use
recieve_notOpened_modification <- merge(x=userID_Jan_not_open, y=january_modified_listings, by="UserID", all=FALSE)
View(recieve_notOpened_modification)

for (i in 1:nrow(recieve_notOpened_modification)) {
  if (recieve_notOpened_modification$ModifiedListing[i] == 1) {  #I want to have only those who did a modification,
    r$SETBIT("EmailsOpenedJanuary", i ,"1")
  }
}

r$BITCOUNT("EmailsOpenedJanuary") #count the userID's that received email on January, they didn't opened it but alt
```

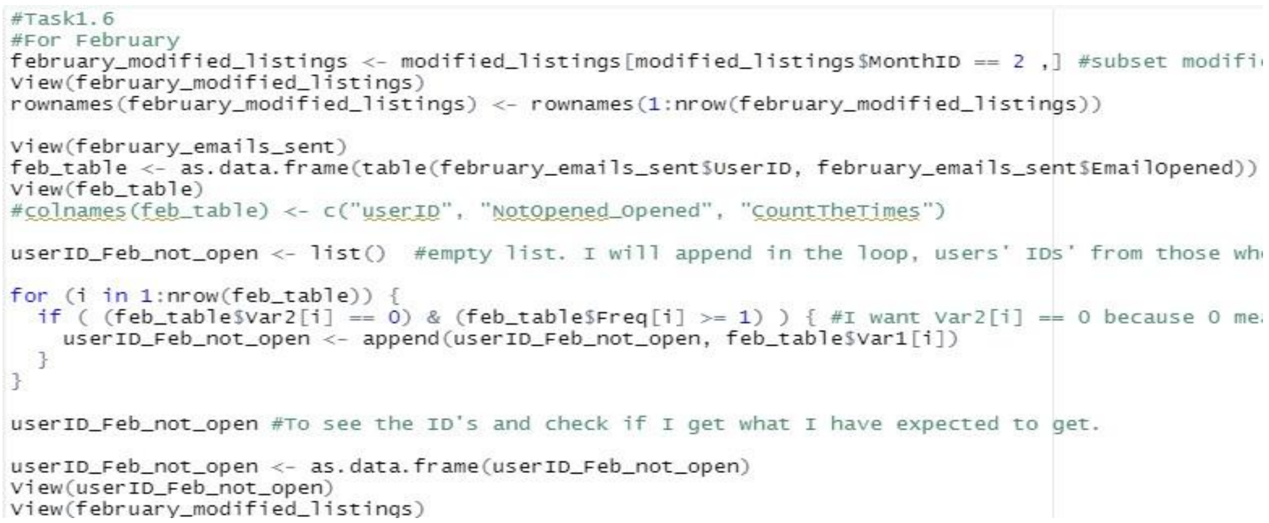
So, I conclude that 2807 userID received an email in January, did not open it but nevertheless they made a modification on their list.



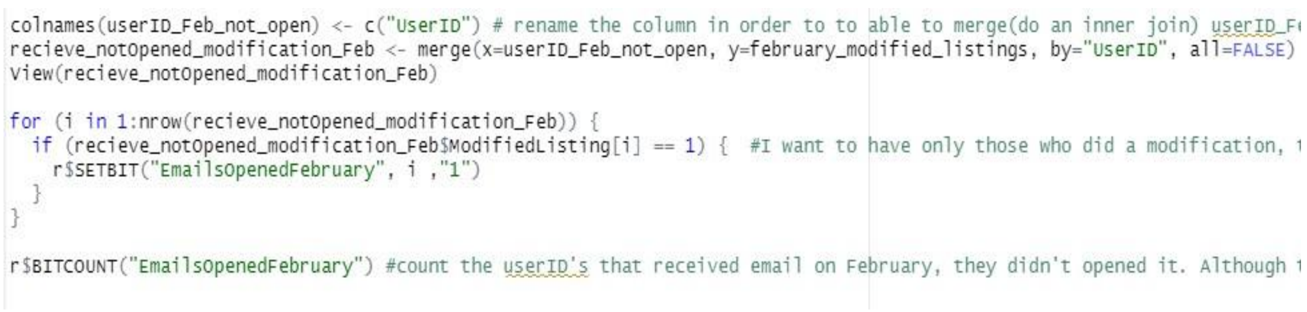
```
> r$BITCOUNT("EmailsOpenedJanuary")
[1] 2807
```

Task 1.6

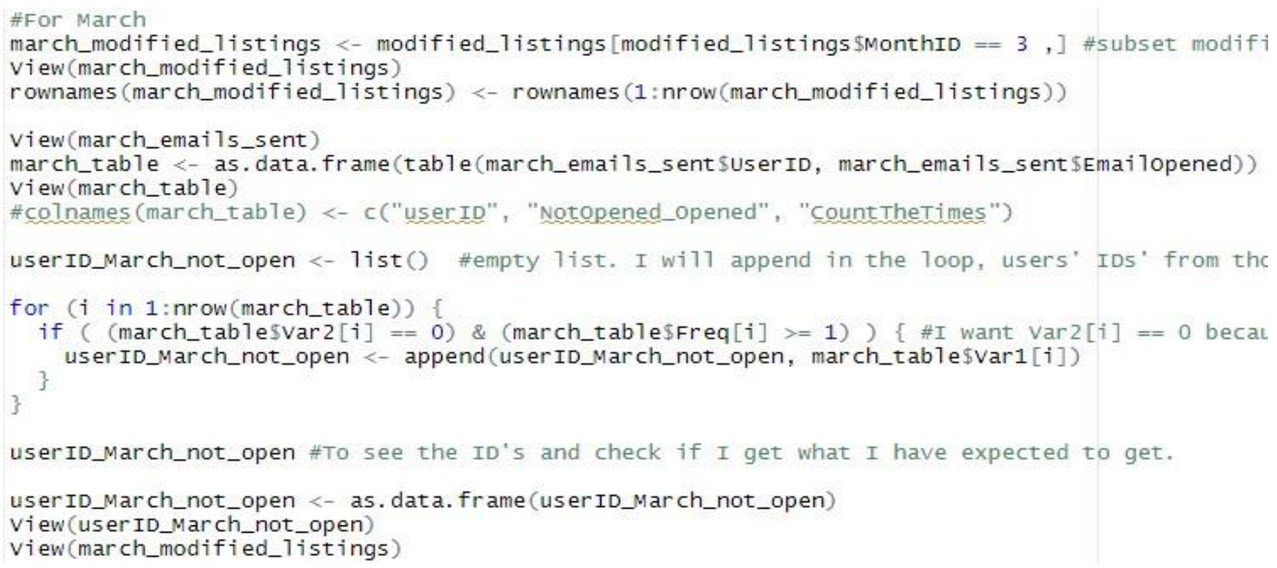
The analysis that should be done in this question concerns the customers who received emails in January or February or March, did not open it but nevertheless they update their list in one of these three specific months.
I already have a Bitmap for January from Task 1.5. Therefore, I will do the same procedure for the months February and March. Finally, with the Bitop OR command command, I count the userIDs that received emails in January or February or March, did not open it but they update their list. Hence, I will "check" if the 3 Bitmaps in each position contain at least one 1.



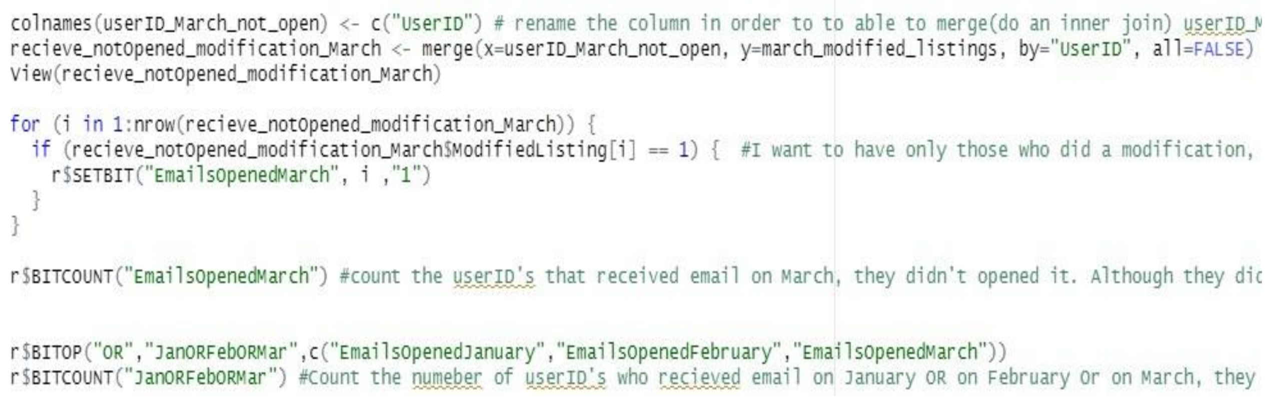
```
#Task1.6
#For February
february_modified_listings <- modified_listings[modified_listings$MonthID == 2 ,] #subset modifi
View(february_modified_listings)
rownames(february_modified_listings) <- rownames(1:nrow(february_modified_listings))

View(february_emails_sent)
feb_table <- as.data.frame(table(february_emails_sent$UserID, february_emails_sent$EmailOpened))
View(feb_table)
#colnames(feb_table) <- c("userID", "NotOpened_Opened", "CountTheTimes")

userID_Feb_not_open <- list()  #empty list. I will append in the loop, users' IDs' from those wh

for (i in 1:nrow(feb_table)) {
  if ( (feb_table$Var2[i] == 0) & (feb_table$Freq[i] >= 1) ) { #I want Var2[i] == 0 because 0 me
    userID_Feb_not_open <- append(userID_Feb_not_open, feb_table$Var1[i])
  }
}

userID_Feb_not_open #To see the ID's and check if I get what I have expected to get.

userID_Feb_not_open <- as.data.frame(userID_Feb_not_open)
View(userID_Feb_not_open)
View(february_modified_listings)
```



```
colnames(userID_Feb_not_open) <- c("UserID") # rename the column in order to to able to merge(do an inner join) userID_F
recieve_notOpened_modification_Feb <- merge(x=userID_Feb_not_open, y=february_modified_listings, by="UserID", all=FALSE)
View(recieve_notOpened_modification_Feb)

for (i in 1:nrow(recieve_notOpened_modification_Feb)) {
  if (recieve_notOpened_modification_Feb$ModifiedListing[i] == 1) {  #I want to have only those who did a modification, 
    r$SETBIT("EmailsOpenedFebruary", i ,"1")
  }
}

r$BITCOUNT("EmailsOpenedFebruary") #count the userID's that received email on February, they didn't opened it. Although 
```

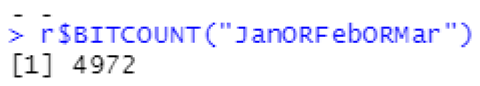


```
#For March
march_modified_listings <- modified_listings[modified_listings$MonthID == 3 ,] #subset modifi
View(march_modified_listings)
rownames(march_modified_listings) <- rownames(1:nrow(march_modified_listings))

View(march_emails_sent)
march_table <- as.data.frame(table(march_emails_sent$UserID, march_emails_sent$EmailOpened))
View(march_table)
#colnames(march_table) <- c("userID", "NotOpened_Opened", "CountTheTimes")

userID_March_not_open <- list()  #empty list. I will append in the loop, users' IDs' from thc

for (i in 1:nrow(march_table)) {
  if ( (march_table$Var2[i] == 0) & (march_table$Freq[i] >= 1) ) { #I want Var2[i] == 0 becau
    userID_March_not_open <- append(userID_March_not_open, march_table$Var1[i])
  }
}

userID_March_not_open #To see the ID's and check if I get what I have expected to get.

userID_March_not_open <- as.data.frame(userID_March_not_open)
View(userID_March_not_open)
View(march_modified_listings)
```



```
colnames(userID_March_not_open) <- c("UserID") # rename the column in order to to able to merge(do an inner join) userID_M
recieve_notOpened_modification_March <- merge(x=userID_March_not_open, y=march_modified_listings, by="UserID", all=FALSE)
View(recieve_notOpened_modification_March)

for (i in 1:nrow(recieve_notOpened_modification_March)) {
  if (recieve_notOpened_modification_March$ModifiedListing[i] == 1) {  #I want to have only those who did a modification,
    r$SETBIT("EmailsOpenedMarch", i ,"1")
  }
}

r$BITCOUNT("EmailsOpenedMarch") #count the userID's that received email on March, they didn't opened it. Although they dic


r$BITOP("OR","JanORFebORMar",c("EmailsOpenedJanuary","EmailsOpenedFebruary","EmailsOpenedMarch"))
r$BITCOUNT("JanORFebORMar") #Count the numeber of userID's who recieved email on January OR on February Or on March, they
```

I conclude that 4972 userID received emails in January or February or March, did not open it but they modidfied their list.

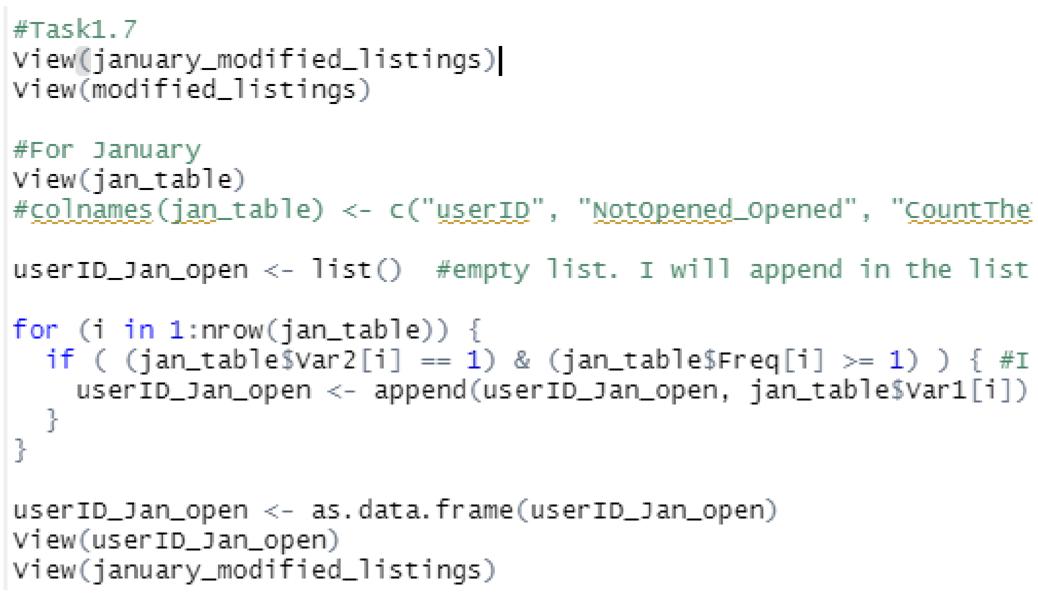


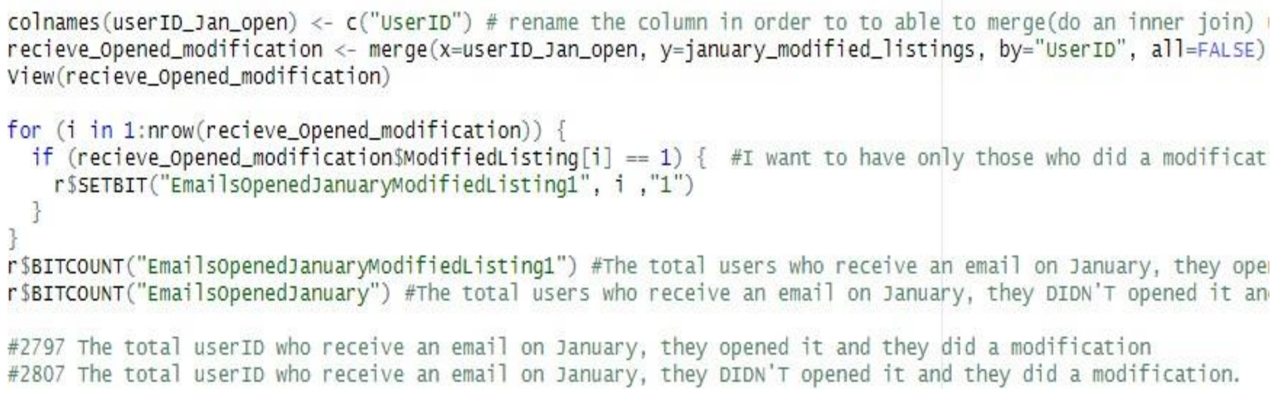
```
> r$BITCOUNT("JanORFebORMar")
[1] 4972
```

Task 1.7

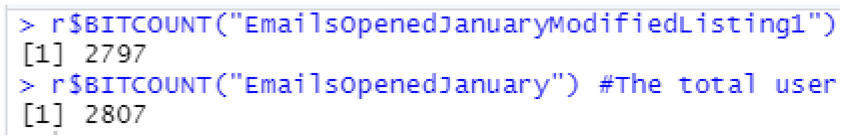
In order to compare and draw a conclusion, I will do the same analysis as I did in Task 1.6 with the difference that I will search for users who received emails in January or February or March, they opened it and modified their list. So, I will compare the results monthly between the Task 1.7 and the Task 1.6 and in total. Accordingly, I will compare for each month, how many users who received email in that month did not open it but made a modification on their list and how many users who received email in that month opened it and made a modification on their list.

For January:

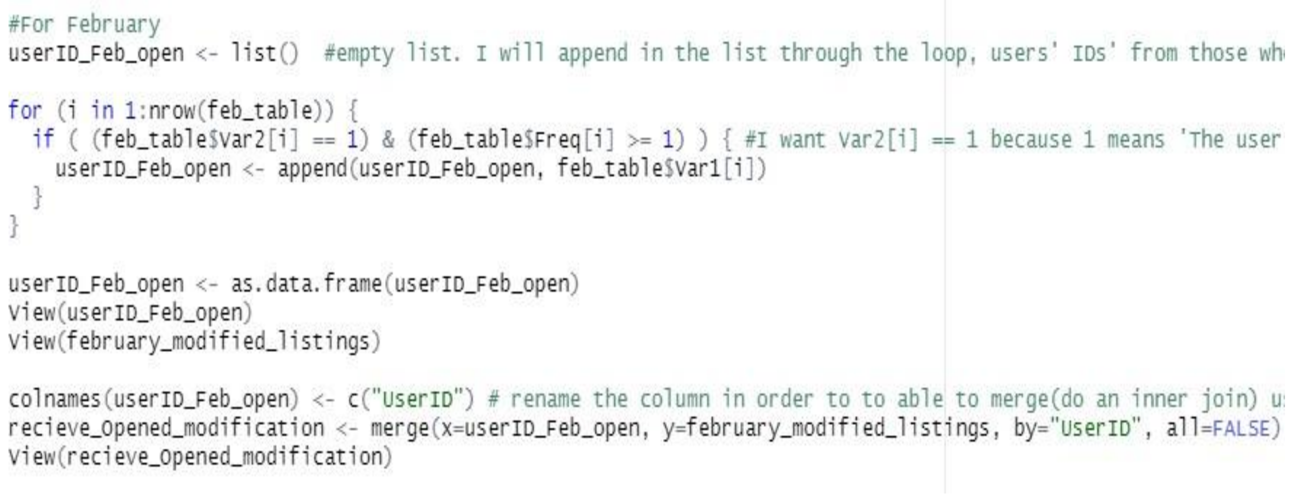

```
#Task1.7
View(january_modified_listings)
View(modified_listings)

#For January
View(jan_table)
#colnames(jan_table) <- c("userID", "NotOpened_Opened", "CountThe

userID_Jan_open <- list()  #empty list. I will append in the list

for (i in 1:nrow(jan_table)) {
  if ( (jan_table$Var2[i] == 1) & (jan_table$Freq[i] >= 1) ) { #I
    userID_Jan_open <- append(userID_Jan_open, jan_table$Var1[i])
  }
}

userID_Jan_open <- as.data.frame(userID_Jan_open)
View(userID_Jan_open)
View(january_modified_listings)
```

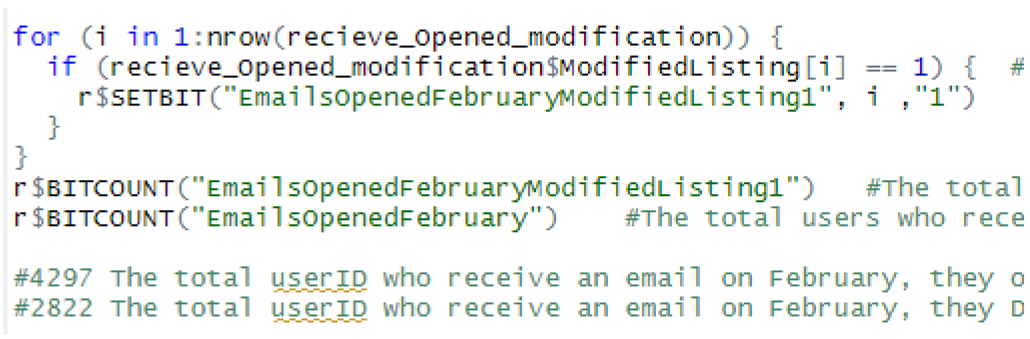

```
colnames(userID_Jan_open) <- c("UserID") # rename the column in order to to able to merge(do an inner join)
recieve_Opened_modification <- merge(x=userID_Jan_open, y=january_modified_listings, by="UserID", all=FALSE)
View(recieve_Opened_modification)

for (i in 1:nrow(recieve_Opened_modification)) {
  if (recieve_Opened_modification$ModifiedListing[i] == 1) {  #I want to have only those who did a modificat
    r$SETBIT("EmailsOpenedJanuaryModifiedListing1", i ,"1")
  }
}
r$BITCOUNT("EmailsOpenedJanuaryModifiedListing1") #The total users who receive an email on January, they ope
r$BITCOUNT("EmailsOpenedJanuary") #The total users who receive an email on January, they DIDN'T opened it an

#2797 The total userID who receive an email on January, they opened it and they did a modification
#2807 The total userID who receive an email on January, they DIDN'T opened it and they did a modification.
```

I notice that the results for January don’t differ much.


```
> r$BITCOUNT("EmailsOpenedJanuaryModifiedListing1")
[1] 2797
> r$BITCOUNT("EmailsOpenedJanuary") #The total user
[1] 2807
```

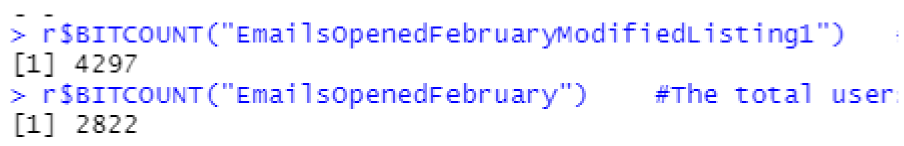
For February:


```
#For February
userID_Feb_open <- list()  #empty list. I will append in the list through the loop, users' IDs' from those wh

for (i in 1:nrow(feb_table)) {
  if ( (feb_table$Var2[i] == 1) & (feb_table$Freq[i] >= 1) ) { #I want Var2[i] == 1 because 1 means 'The user
    userID_Feb_open <- append(userID_Feb_open, feb_table$Var1[i])
  }
}

userID_Feb_open <- as.data.frame(userID_Feb_open)
View(userID_Feb_open)
View(february_modified_listings)

colnames(userID_Feb_open) <- c("UserID") # rename the column in order to to able to merge(do an inner join) u
recieve_Opened_modification <- merge(x=userID_Feb_open, y=february_modified_listings, by="UserID", all=FALSE)
View(recieve_Opened_modification)
```



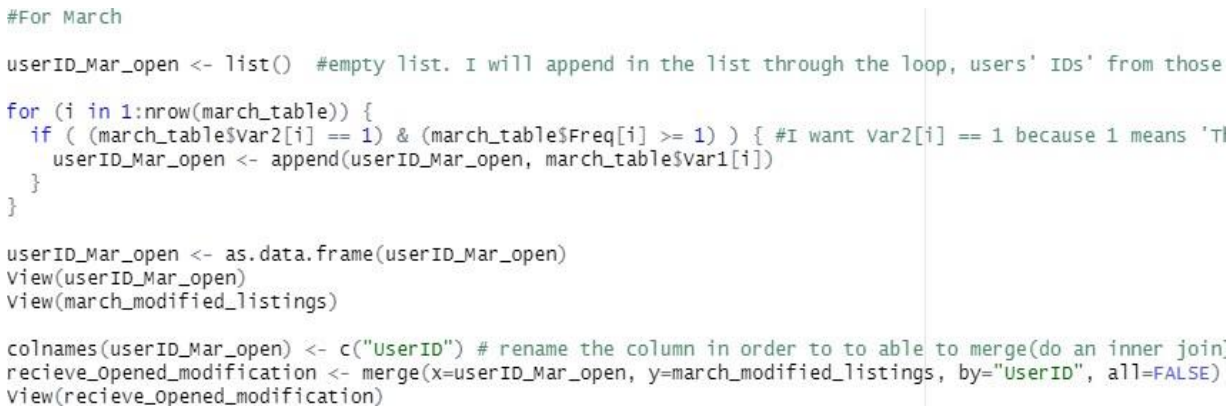
```
for (i in 1:nrow(recieve_Opened_modification)) {
  if (recieve_Opened_modification$ModifiedListing[i] == 1) {   #
    r$SETBIT("EmailsOpenedFebruaryModifiedListing1", i ,"1")
  }
}
r$BITCOUNT("EmailsOpenedFebruaryModifiedListing1")    #The total
r$BITCOUNT("EmailsOpenedFebruary")     #The total users who rece

#4297 The total userID who receive an email on February, they o
#2822 The total userID who receive an email on February, they D
```

I notice for January that the users who received email, opened it and made a modification are more than the users who did not open it and made a modification.

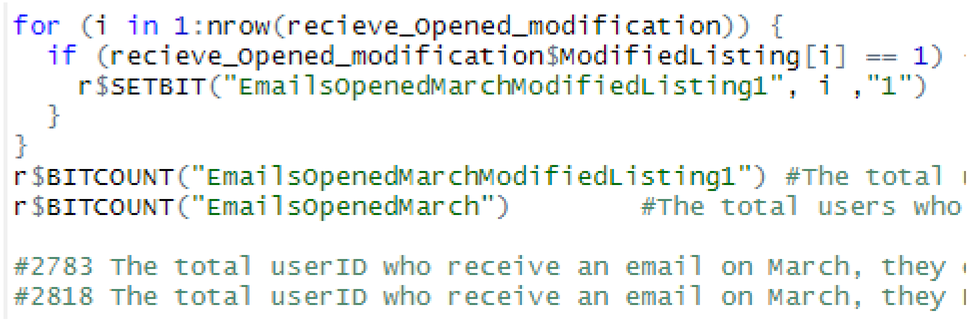


```
> r$BITCOUNT("EmailsOpenedFebruaryModifiedListing1")
[1] 4297
> r$BITCOUNT("EmailsOpenedFebruary")     #The total user
[1] 2822
```

For March:

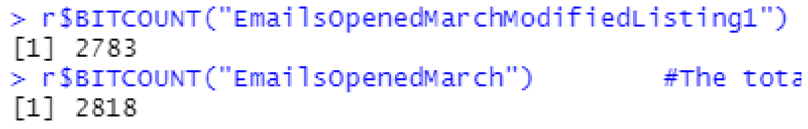


```
#For March

userID_Mar_open <- list()  #empty list. I will append in the list through the loop, users' IDs' from those

for (i in 1:nrow(march_table)) {
  if ( (march_table$Var2[i] == 1) & (march_table$Freq[i] >= 1) ) { #I want Var2[i] == 1 because 1 means 'T
    userID_Mar_open <- append(userID_Mar_open, march_table$Var1[i])
  }
}

userID_Mar_open <- as.data.frame(userID_Mar_open)
View(userID_Mar_open)
View(march_modified_listings)

colnames(userID_Mar_open) <- c("UserID") # rename the column in order to to able to merge(do an inner join
recieve_Opened_modification <- merge(x=userID_Mar_open, y=march_modified_listings, by="UserID", all=FALSE)
View(recieve_Opened_modification)
```

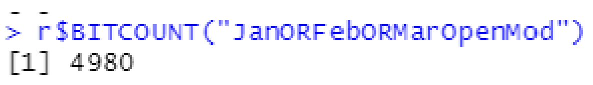


```
for (i in 1:nrow(recieve_Opened_modification)) {
  if (recieve_Opened_modification$ModifiedListing[i] == 1)
    r$SETBIT("EmailsOpenedMarchModifiedListing1", i ,"1")
  }
}
r$BITCOUNT("EmailsOpenedMarchModifiedListing1") #The total
r$BITCOUNT("EmailsOpenedMarch")          #The total users who

#2783 The total userID who receive an email on March, they
#2818 The total userID who receive an email on March, they
```

I notice that the results for March don’t differ much.

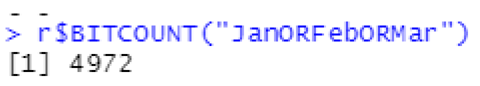


```
> r$BITCOUNT("EmailsOpenedMarchModifiedListing1")
[1] 2783
> r$BITCOUNT("EmailsOpenedMarch")        #The tota
[1] 2818
```



```
> r$BITCOUNT("JanORFebORMarOpenMod")
[1] 4980
```

By executing the Bitop OR command, I will count the userIDs who received emails
in January or in February or in March, they opened them and update their list.
Consequently, I will "check" if the 3 new Bitmaps in each position contain at least
one 1.
Comparing with the result from Task 1.6, I notice that there is no difference between them.



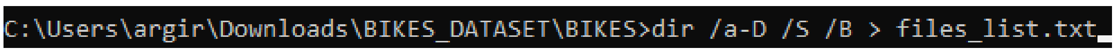
```
> r$BITCOUNT("JanORFebORMar")
[1] 4972
```

Consequently, I conclude that there should be a different approach in the way users
are informed as almost ½ of them does not even read the emails.

Mongo

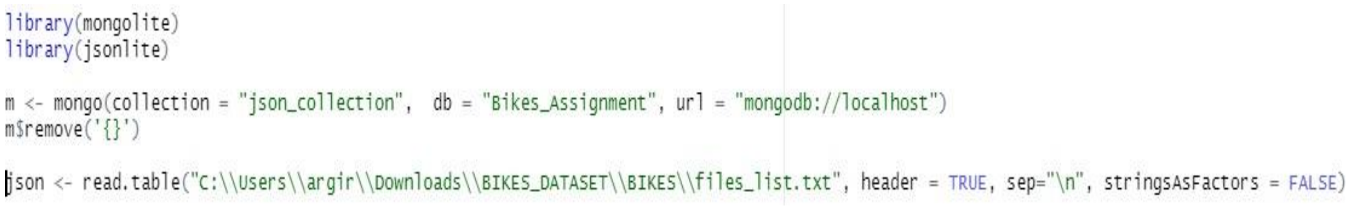
Task 2.1

Firstly, I create the files_list in the folder that contains the json files so that the
changes I will make be executed much faster and automatically in all json files.



```
C:\Users\argir\Downloads\BIKES_DATASET\BIKES>dir /a-D /S /B > files_list.txt
```

Then I read the files_list and connect the programming language r to Mongo through the mongolite
library with the mongo () command where I create a database called Bikes_Assignment and a
collection called json_collection.



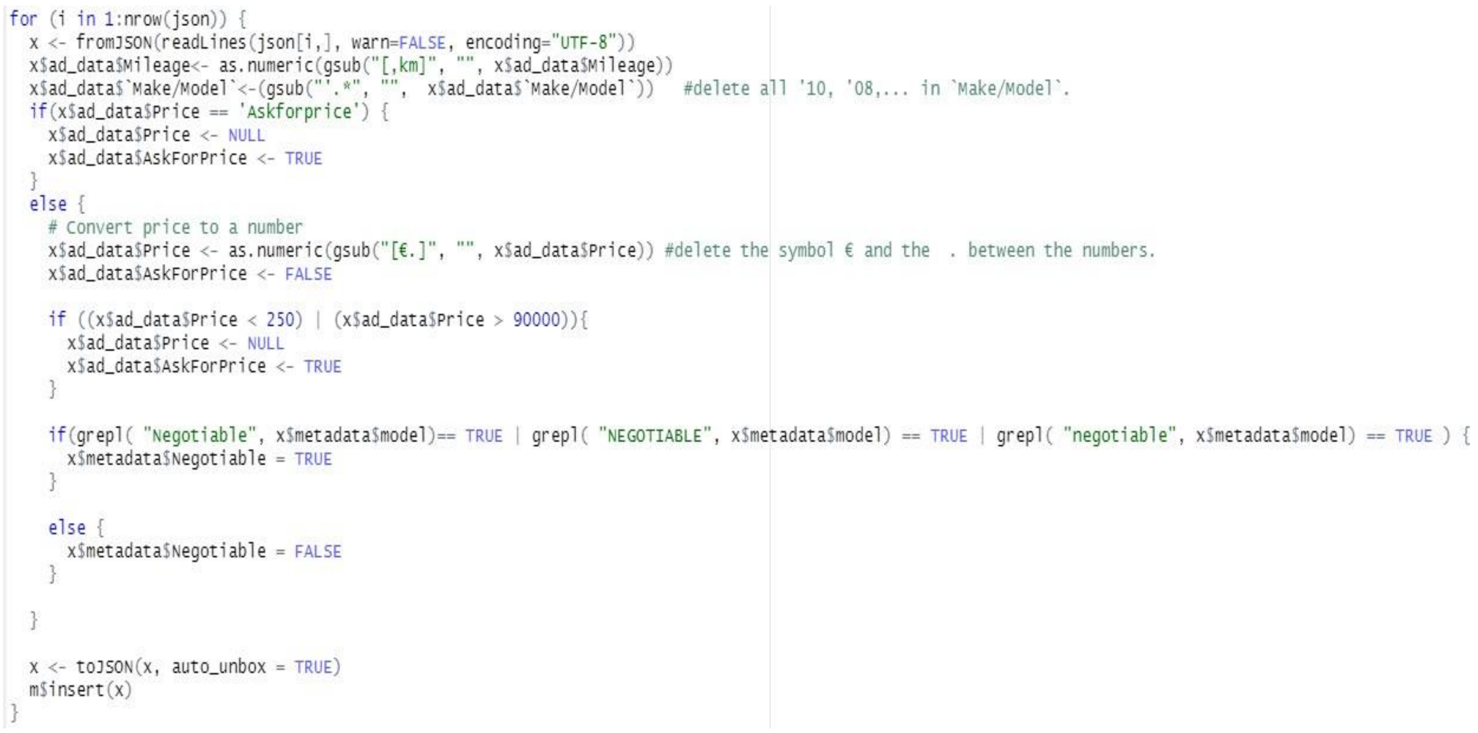
```
library(mongolite)
library(jsonlite)

m <- mongo(collection = "json_collection",  db = "Bikes_Assignment", url = "mongodb://localhost")
m$remove('{}')

json <- read.table("C:\\Users\\argir\\Downloads\\BIKES_DATASET\\BIKES\\files_list.txt", header = TRUE, sep="\n", stringsAsFactors = FALSE)
```

The next step is to clear the data (the contents of the json files) and import them into my database in
order to proceed with the analysis.
In order to proceed with the inspection and with modifications to each json file, I will
use the for command and the command fromJSON (readLines). By observing the
json files in the visual studio, I notice that the Mileage field contains letters (km) and
punctuation (comma) which must be removed. Also, from sting type, they should be
converted to numeric in order to be able to perform Task 2.10. I also notice that in
the field `Make / Model` besides the model, the date have been written by using the
symbol ‘ and the last 2 numbers of the year ('. *) which I will have to remove so as to
answer task 2.6, task 2.7 and task 2.8. Thus, I remove what exists from the
punctuation mark onwards.

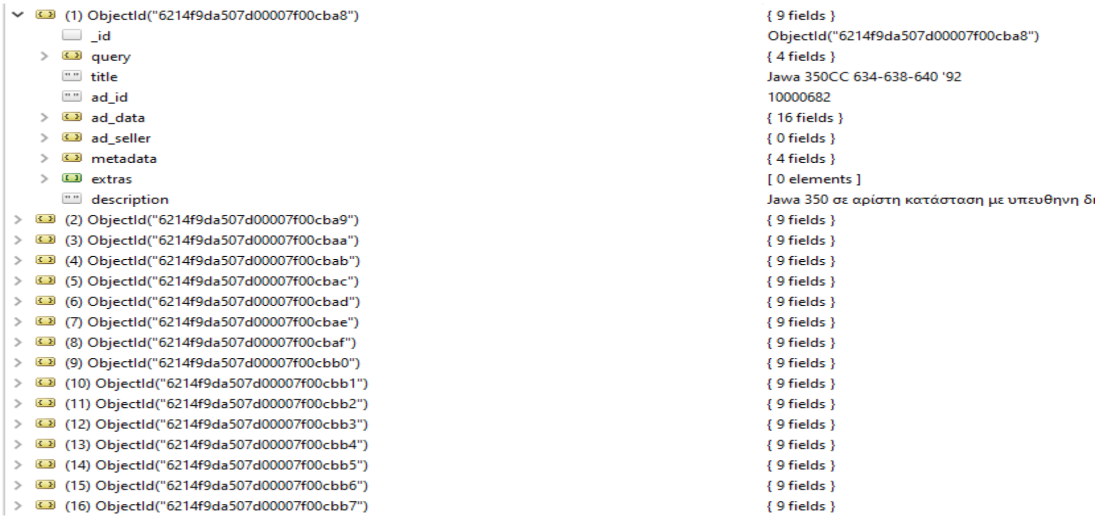
The Price field contains a symbol (€) and a full stop (dot) between the numbers
which must also be removed. Also, from sting type they must
be converted to numeric, so as analysis and mathematical operations to be
performed. I also notice that there are many ads in which the price is quite low.
That's why the ads that have price less than 250 euros or

more than 90000 euros (because some motorcycles may be collective), I consider them incorrect and change the price to NA and change the AskForPrice field in ad.data as TRUE.Consequently, I can answer Tasks 2.3, 2.4.

Lastly, in the metadata field and more specifically in the model field, where there is the word Negotiable or NEGOTIABLE or negotiable in the metadata field and more specifically in the Negotiable field I put the True value to answer Task 2.5.



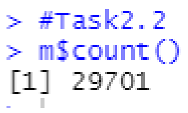
```r
for (i in 1:nrow(json)) {
  x <- fromJSON(readLines(json[i,], warn=FALSE, encoding="UTF-8"))
  x$ad_data$Mileage<- as.numeric(gsub("[,km]", "", x$ad_data$Mileage))
  x$ad_data$`Make/Model`<-(gsub("'.*", "",  x$ad_data$`Make/Model`))   #delete all '10, '08,... in `Make/Model`.
  if(x$ad_data$Price == 'Askforprice') {
    x$ad_data$Price <- NULL
    x$ad_data$AskForPrice <- TRUE
  }
  else {
    # Convert price to a number
    x$ad_data$Price <- as.numeric(gsub("[€.]", "", x$ad_data$Price)) #delete the symbol € and the  . between the numbers.
    x$ad_data$AskForPrice <- FALSE

    if ((x$ad_data$Price < 250) | (x$ad_data$Price > 90000)){
      x$ad_data$Price <- NULL
      x$ad_data$AskForPrice <- TRUE
    }

    if(grepl( "Negotiable", x$metadata$model)== TRUE | grepl( "NEGOTIABLE", x$metadata$model) == TRUE | grepl( "negotiable", x$metadata$model) == TRUE ) {
      x$metadata$Negotiable = TRUE
    }

    else {
      x$metadata$Negotiable = FALSE
    }

  }

  x <- toJSON(x, auto_unbox = TRUE)
  m$insert(x)
}
```



```
v (1) ObjectId("6214f9da507d00007f00cba8")      { 9 fields }
    _id                                          ObjectId("6214f9da507d00007f00cba8")
  > query                                        { 4 fields }
    title                                        Jawa 350CC 634-638-640 '92
    ad_id                                        10000682
  > ad_data                                      { 16 fields }
  > ad_seller                                    { 0 fields }
  > metadata                                     { 4 fields }
  > extras                                       [ 0 elements ]
    description                                  Jawa 350 σε αρίστη κατάσταση με υπευθηνη δι
> (2) ObjectId("6214f9da507d00007f00cba9")      { 9 fields }
> (3) ObjectId("6214f9da507d00007f00cbaa")      { 9 fields }
> (4) ObjectId("6214f9da507d00007f00cbab")      { 9 fields }
> (5) ObjectId("6214f9da507d00007f00cbac")      { 9 fields }
> (6) ObjectId("6214f9da507d00007f00cbad")      { 9 fields }
> (7) ObjectId("6214f9da507d00007f00cbae")      { 9 fields }
> (8) ObjectId("6214f9da507d00007f00cbaf")      { 9 fields }
> (9) ObjectId("6214f9da507d00007f00cbb0")      { 9 fields }
> (10) ObjectId("6214f9da507d00007f00cbb1")     { 9 fields }
> (11) ObjectId("6214f9da507d00007f00cbb2")     { 9 fields }
> (12) ObjectId("6214f9da507d00007f00cbb3")     { 9 fields }
> (13) ObjectId("6214f9da507d00007f00cbb4")     { 9 fields }
> (14) ObjectId("6214f9da507d00007f00cbb5")     { 9 fields }
> (15) ObjectId("6214f9da507d00007f00cbb6")     { 9 fields }
> (16) ObjectId("6214f9da507d00007f00cbb7")     { 9 fields }
```

Task 2.2

In order to count the ads, I just execute the count command in the collection.

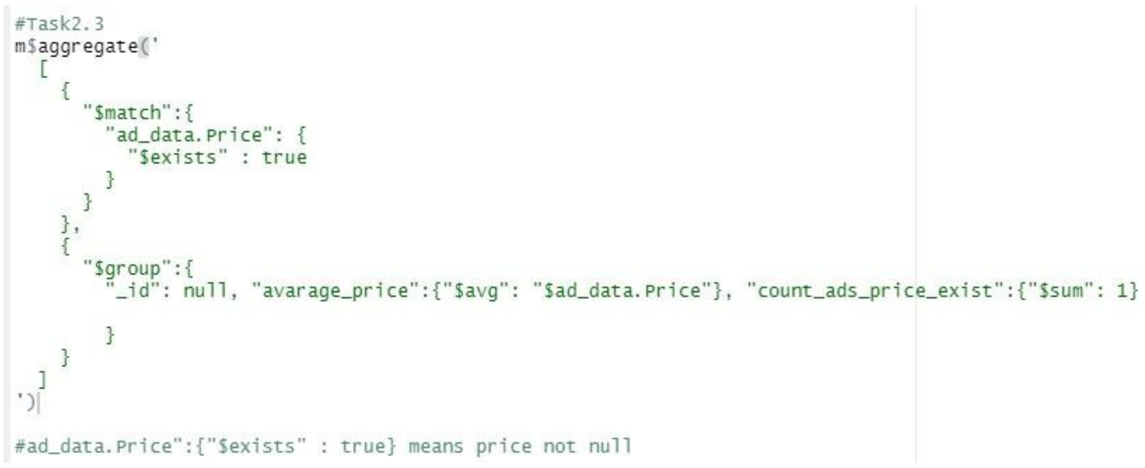


```
> #Task2.2
> m$count()
[1] 29701
```

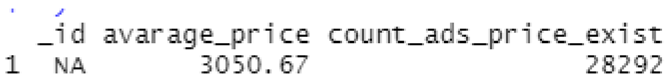
29701 is the number of the ads in the collection.

Task 2.3

The average price is 3050.67 euros. The ads I considered to calculate the average price were 28292.



```
#Task2.3
m$aggregate('
   [
     {
       "$match":{
         "ad_data.Price": {
           "$exists" : true
         }
       }
     },
     {
       "$group":{
         "_id": null, "avarage_price":{"$avg": "$ad_data.Price"}, "count_ads_price_exist":{"$sum": 1}

         }
     }
   ]
')

#ad_data.Price":{"$exists" : true} means price not null
```

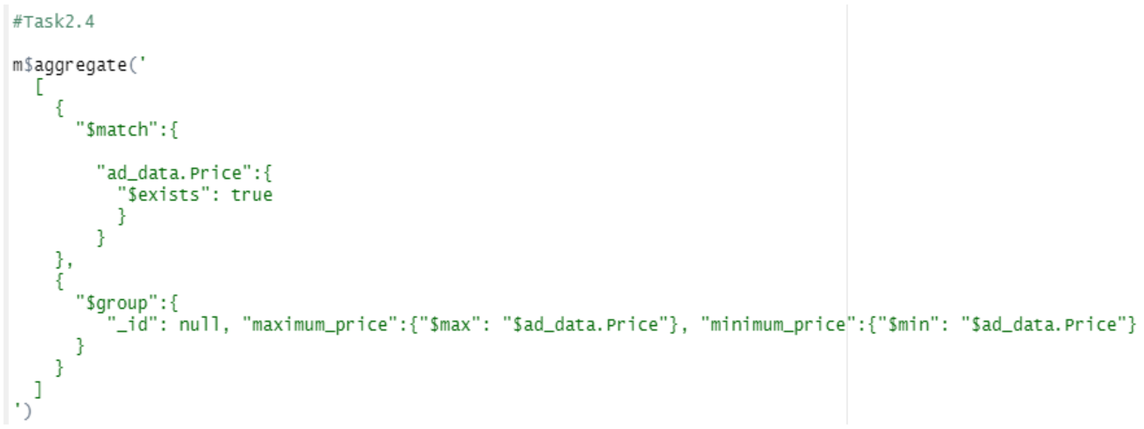


```
  _id avarage_price count_ads_price_exist
1  NA       3050.67                 28292
```

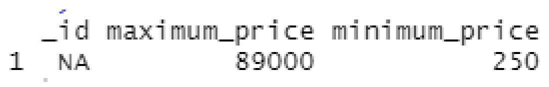
I observe that the result is different form the result of the Task 2.2. This occurs from the fact that in the ads which have price less than 250 euro or more than 90000 euro, we replace the price with NA value. Because in the “query” I asked for the ads that have price, I received 28292 out of 29701. This means that 1409 ads have NA in price field after the cleaning procedure.

Task 2.4

The minimum and maximum price observed in the ads after clearing the data is 250 euros and 89000 euros respectively.



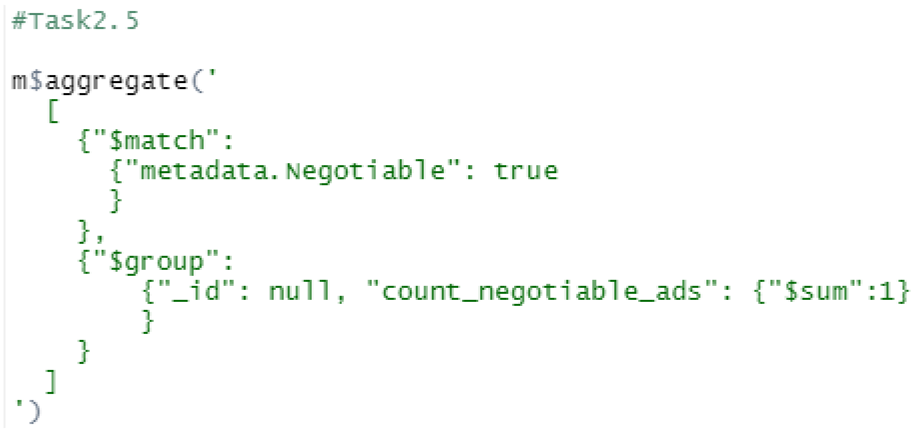
```
#Task2.4

m$aggregate('
  [
    {
      "$match":{

        "ad_data.Price":{
          "$exists": true
          }
        }
    },
    {
      "$group":{
        "_id": null, "maximum_price":{"$max": "$ad_data.Price"}, "minimum_price":{"$min": "$ad_data.Price"}
      }
    }
  ]
')
```



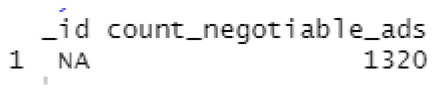
```
  _id maximum_price minimum_price
1  NA         89000           250
```

Task 2.5

To find the number of ads that have a Negotiable price, I count the ads where in the metadata field and more specifically in the Negotiable field they have the value true.



```
#Task2.5

m$aggregate('
  [
    {"$match":
      {"metadata.Negotiable": true
      }
    },
    {"$group":
        {"_id": null, "count_negotiable_ads": {"$sum":1}
        }
    }
  ]
')
```

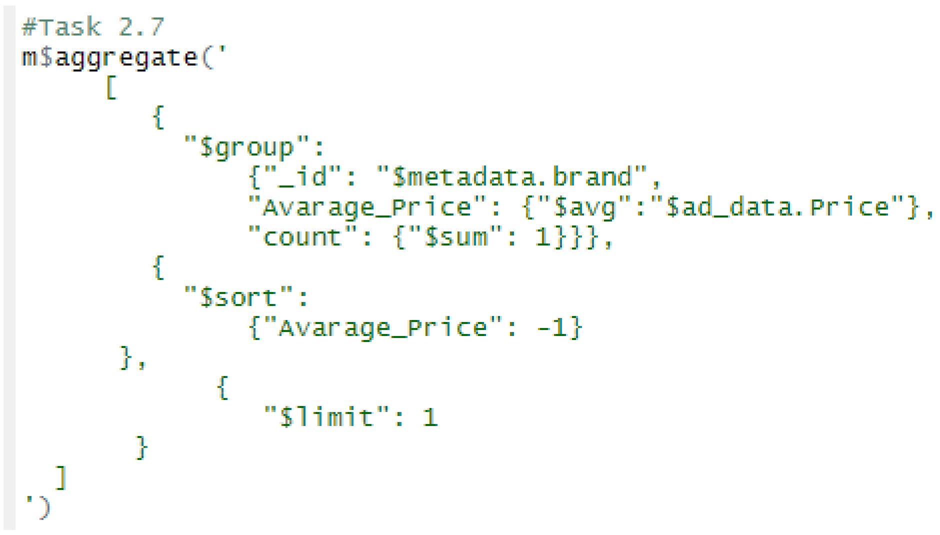


```
  _id count_negotiable_ads
1  NA                 1320
```

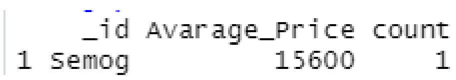
The number of ads which have Negotiable price is 1320.

Task 2.7

The brand with the highest average price is the Semog brand with an average price of 15,600 euros.



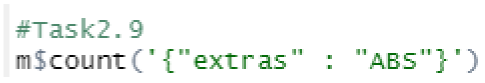
```
#Task 2.7
m$aggregate('
     [
        {
          "$group":
              {"_id": "$metadata.brand",
              "Avarage_Price": {"$avg":"$ad_data.Price"},
              "count": {"$sum": 1}}},
        {
          "$sort":
              {"Avarage_Price": -1}
      },
            {
               "$limit": 1
       }
  ]
')
```



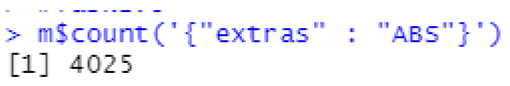
```
    _id Avarage_Price count
1 Semog         15600     1
```

Task 2.9

In order to count the motorcycles that have ABS as an extra, I just execute count command in the extra field for the ABS.



```
#Task2.9
m$count('{"extras" : "ABS"}')
```



```
> m$count('{"extras" : "ABS"}')
[1] 4025
```