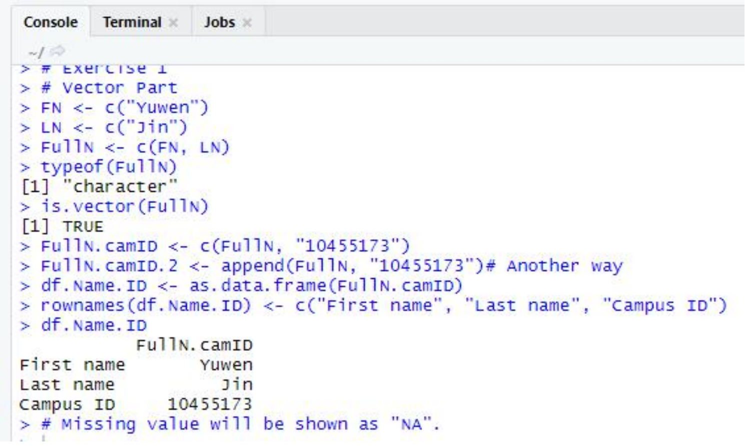
Exercise 1:

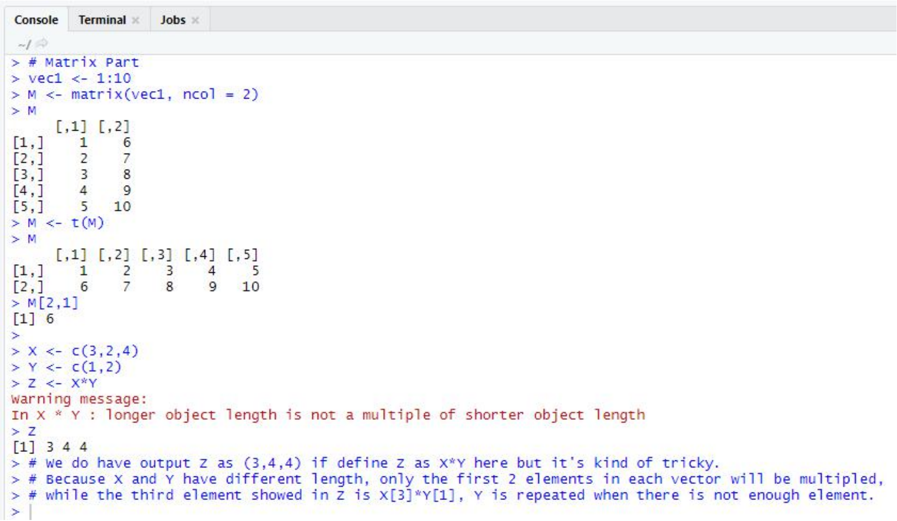
Vector part:



```
> # Exercise 1
> # Vector Part
> FN <- c("Yuwen")
> LN <- c("Jin")
> FullN <- c(FN, LN)
> typeof(FullN)
[1] "character"
> is.vector(FullN)
[1] TRUE
> FullN.camID <- c(FullN, "10455173")
> FullN.camID.2 <- append(FullN, "10455173")# Another way
> df.Name.ID <- as.data.frame(FullN.camID)
> rownames(df.Name.ID) <- c("First name", "Last name", "Campus ID")
> df.Name.ID
           FullN.camID
First name       Yuwen
Last name          Jin
Campus ID     10455173
> # Missing value will be shown as "NA".
```

- Missing values are shown as “NA”.

Matrix Part :



```
> # Matrix Part
> vec1 <- 1:10
> M <- matrix(vec1, ncol = 2)
> M
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
> M <- t(M)
> M
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
> M[2,1]
[1] 6
>
> X <- c(3,2,4)
> Y <- c(1,2)
> Z <- X*Y
Warning message:
In X * Y : longer object length is not a multiple of shorter object length
> Z
[1] 3 4 4
> # We do have output Z as (3,4,4) if define Z as X*Y here but it's kind of tricky.
> # Because X and Y have different length, only the first 2 elements in each vector will be multipled,
> # while the third element showed in Z is X[3]*Y[1], Y is repeated when there is not enough element.
> |
```

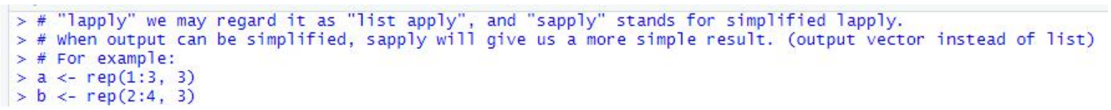
We do have output Z as (3,4,4) if define Z as X*Y here but it's kind of tricky.

Because X and Y have different length, only the first elements in each vector will be multiplied, while the third element showed in Z is X[3]*Y[1], Y is repeated when there is not enough element.

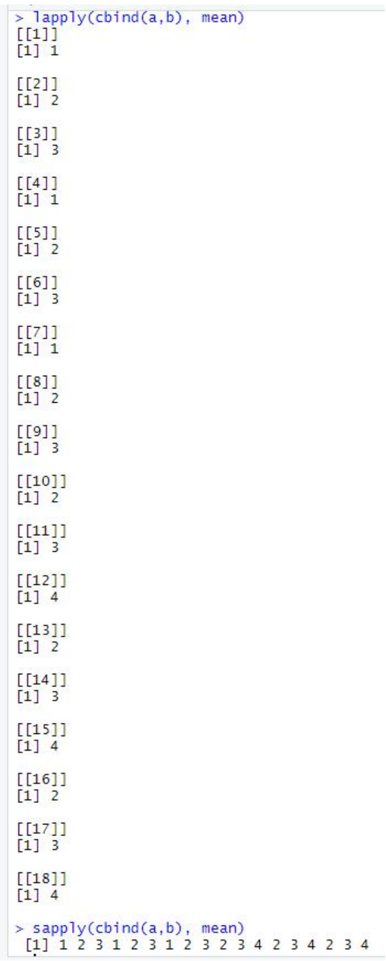
Function Part:

"With()" means limit operation to certain range. For example, with(SIT, students) means call the "students" items in data "SIT".
"By()" function applies a function to a data frame split by factors

"lapply" we may regard it as "list apply", and "sapply" stands for simplified "lapply" .
When output can be simplified, sapply will give us a more simple result. (output vector instead of list)

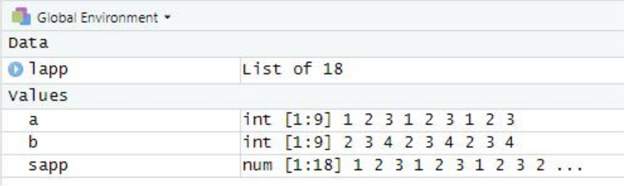


```
> # "lapply" we may regard it as "list apply", and "sapply" stands for simplified lapply.
> # When output can be simplified, sapply will give us a more simple result. (output vector instead of list)
> # For example:
> a <- rep(1:3, 3)
> b <- rep(2:4, 3)
```



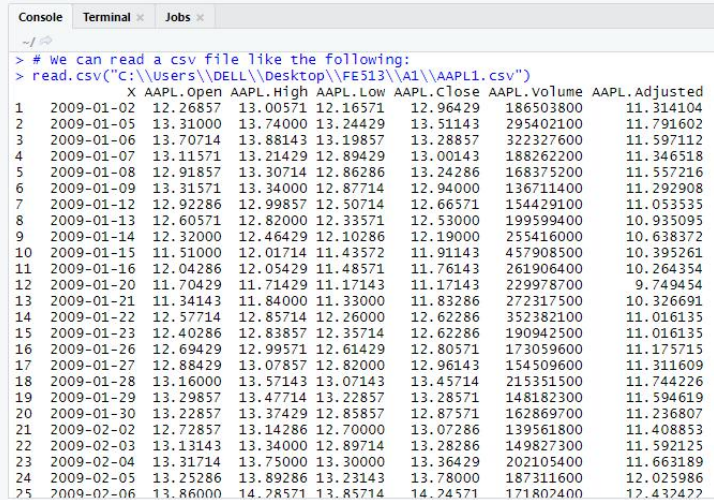
```
> lapply(cbind(a,b), mean)
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

[[4]]
[1] 1

[[5]]
[1] 2

[[6]]
[1] 3

[[7]]
[1] 1

[[8]]
[1] 2

[[9]]
[1] 3

[[10]]
[1] 2

[[11]]
[1] 3

[[12]]
[1] 4

[[13]]
[1] 2

[[14]]
[1] 3

[[15]]
[1] 4

[[16]]
[1] 2

[[17]]
[1] 3

[[18]]
[1] 4

> sapply(cbind(a,b), mean)
 [1] 1 2 3 1 2 3 1 2 3 2 3 4 2 3 4 2 3 4
```

If I define lapp and sapp, from environment we see the differences:



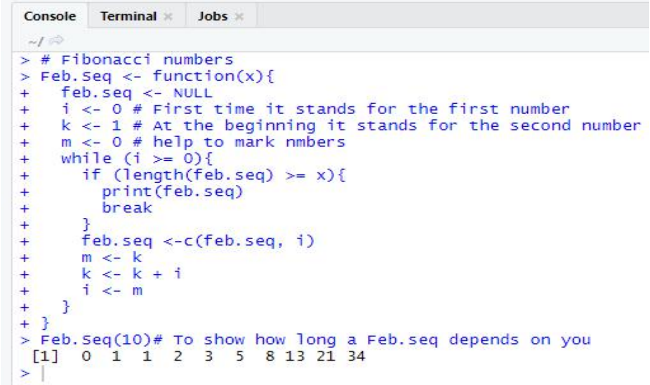
```
Data
 lapp          List of 18
Values
  a            int [1:9] 1 2 3 1 2 3 1 2 3
  b            int [1:9] 2 3 4 2 3 4 2 3 4
  sapp         num [1:18] 1 2 3 1 2 3 1 2 3 2 ...
```

Read csv:

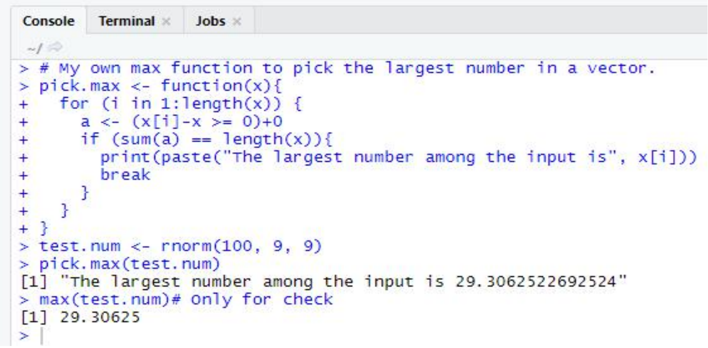


```
> # We can read a csv file like the following:
> read.csv("C:\\Users\\DELL\\Desktop\\FE513\\A1\\AAPL1.csv")
            X AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
1  2009-01-02  12.26857  13.00571 12.16571   12.96429   186503800     11.314104
2  2009-01-05  13.31000  13.74000 13.24429   13.51143   295402100     11.791602
3  2009-01-06  13.70714  13.88143 13.19857   13.28857   322327600     11.597112
4  2009-01-07  13.11571  13.21429 12.89429   13.00143   188262200     11.346518
5  2009-01-08  12.91857  13.30714 12.86286   13.24286   168375200     11.557216
6  2009-01-09  13.31571  13.34000 12.87714   12.94000   136711400     11.292908
7  2009-01-12  12.92286  12.99857 12.50714   12.66571   154429100     11.053535
8  2009-01-13  12.60571  12.82000 12.33571   12.53000   199599400     10.935095
9  2009-01-14  12.32000  12.46429 12.10286   12.19000   255416000     10.638372
10 2009-01-15  11.51000  12.01714 11.43572   11.91143   457908500     10.395261
11 2009-01-16  12.04286  12.05429 11.48571   11.76143   261906400     10.264354
12 2009-01-20  11.70429  11.71429 11.17143   11.17143   229978700      9.749454
13 2009-01-21  11.34143  11.84000 11.33000   11.83286   272317500     10.326691
14 2009-01-22  12.57714  12.85714 12.26000   12.62286   352382100     11.016135
15 2009-01-23  12.40286  12.83857 12.35714   12.62286   190942500     11.016135
16 2009-01-26  12.69429  12.99571 12.61429   12.80571   173059600     11.175715
17 2009-01-27  12.88429  13.07857 12.82000   12.96143   154509600     11.311609
18 2009-01-28  13.16000  13.57143 13.07143   13.45714   215351500     11.744226
19 2009-01-29  13.29857  13.47714 13.22857   13.28571   148182300     11.594619
20 2009-01-30  13.22857  13.37429 12.85857   12.87571   162869700     11.236807
21 2009-02-02  12.72857  13.14286 12.70000   13.07286   139561800     11.408853
22 2009-02-03  13.13143  13.34000 12.89714   13.28286   149827300     11.592125
23 2009-02-04  13.31714  13.75000 13.30000   13.36429   202105400     11.663189
24 2009-02-05  13.25286  13.89286 13.23143   13.78000   187311600     12.025986
25 2009-02-06  13.86000  14.28571 13.85714   14.24571   171802400     12.432422
```

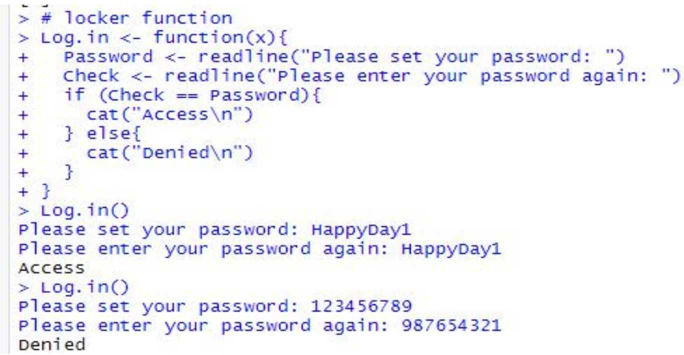
Application:



```
> # Fibonacci numbers
> Feb.seq <- function(x){
+   feb.seq <- NULL
+   i <- 0 # First time it stands for the first number
+   k <- 1 # At the beginning it stands for the second number
+   m <- 0 # help to mark nmbers
+   while (i >= 0){
+     if (length(feb.seq) >= x){
+       print(feb.seq)
+       break
+     }
+     feb.seq <-c(feb.seq, i)
+     m <- k
+     k <- k + i
+     i <- m
+   }
+ }
> Feb.seq(10)# To show how long a Feb.seq depends on you
 [1]  0  1  1  2  3  5  8 13 21 34
> 
```

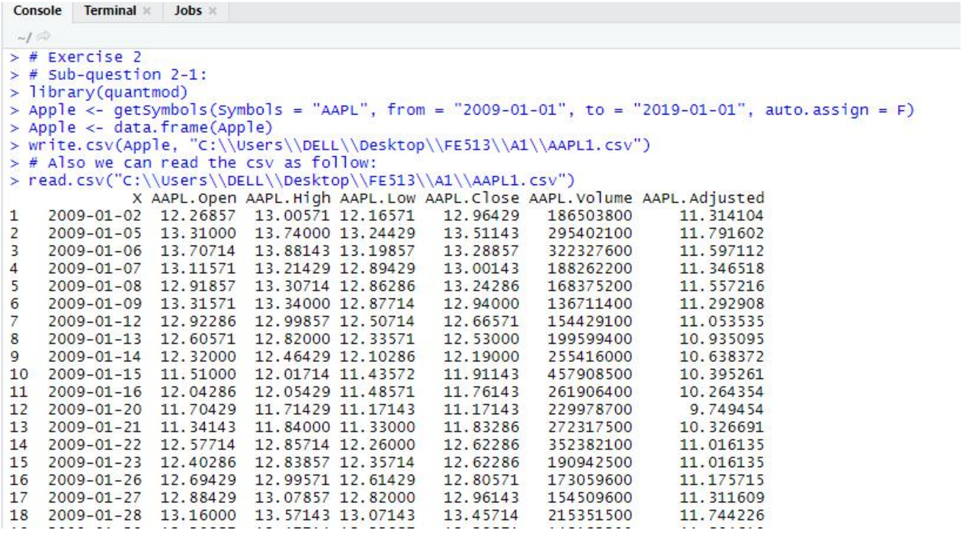


```
> # My own max function to pick the largest number in a vector.
> pick.max <- function(x){
+   for (i in 1:length(x)) {
+     a <- (x[i]-x >= 0)+0
+     if (sum(a) == length(x)){
+       print(paste("The largest number among the input is", x[i]))
+       break
+     }
+   }
+ }
> test.num <- rnorm(100, 9, 9)
> pick.max(test.num)
[1] "The largest number among the input is 29.3062522692524"
> max(test.num)# Only for check
[1] 29.30625
> 
```

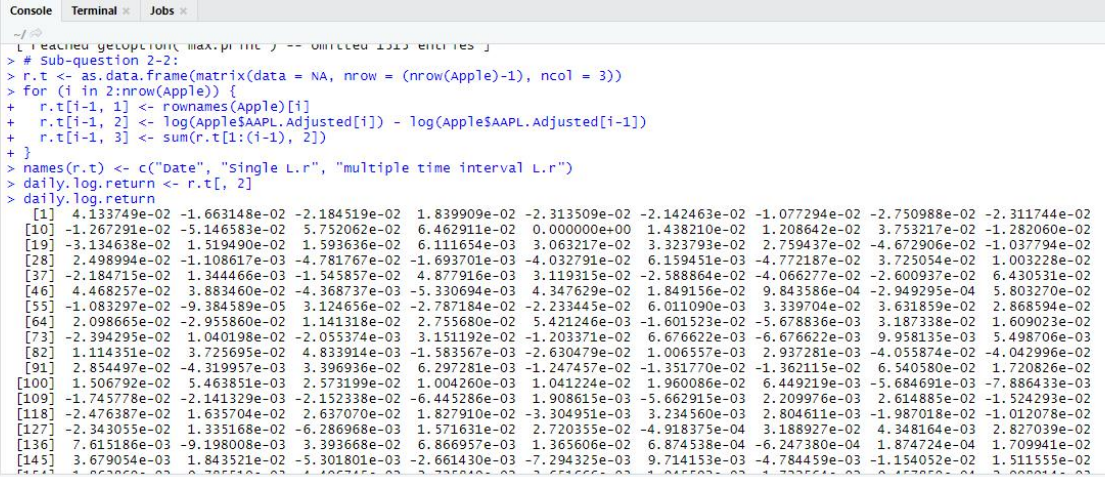


```
> # locker function
> Log.in <- function(x){
+   Password <- readline("Please set your password: ")
+   Check <- readline("Please enter your password again: ")
+   if (Check == Password){
+     cat("Access\n")
+   } else{
+     cat("Denied\n")
+   }
+ }
> Log.in()
Please set your password: HappyDay1
Please enter your password again: HappyDay1
Access
> Log.in()
Please set your password: 123456789
Please enter your password again: 987654321
Denied
```

Exercise 2:

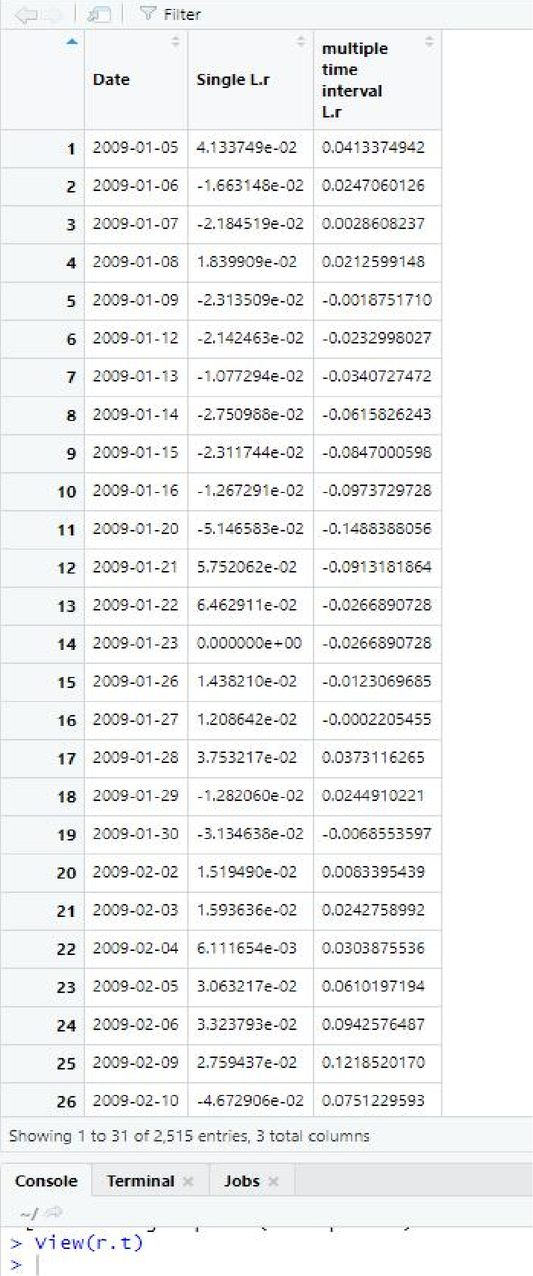


```
> # Exercise 2
> # Sub-question 2-1:
> library(quantmod)
> Apple <- getSymbols(Symbols = "AAPL", from = "2009-01-01", to = "2019-01-01", auto.assign = F)
> Apple <- data.frame(Apple)
> write.csv(Apple, "C:\\Users\\DELL\\Desktop\\FE513\\A1\\AAPL1.csv")
> # Also we can read the csv as follow:
> read.csv("C:\\Users\\DELL\\Desktop\\FE513\\A1\\AAPL1.csv")
              X AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume AAPL.Adjusted
1    2009-01-02  12.26857  13.00571 12.16571   12.96429   186503800     11.314104
2    2009-01-05  13.31000  13.74000 13.24429   13.51143   295402100     11.791602
3    2009-01-06  13.70714  13.88143 13.19857   13.28857   322327600     11.597112
4    2009-01-07  13.11571  13.21429 12.89429   13.00143   188262200     11.346518
5    2009-01-08  12.91857  13.30714 12.86286   13.24286   168375200     11.557216
6    2009-01-09  13.31571  13.34000 12.87714   12.94000   136711400     11.292908
7    2009-01-12  12.92286  12.99857 12.50714   12.66571   154429100     11.053535
8    2009-01-13  12.60571  12.82000 12.33571   12.53000   199599400     10.935095
9    2009-01-14  12.32000  12.46429 12.10286   12.19000   255416000     10.638372
10   2009-01-15  11.51000  12.01714 11.43572   11.91143   457908500     10.395261
11   2009-01-16  12.04286  12.05429 11.48571   11.76143   261906400     10.264354
12   2009-01-20  11.70429  11.71429 11.17143   11.17143   229978700      9.749454
13   2009-01-21  11.34143  11.84000 11.33000   11.83286   272317500     10.326691
14   2009-01-22  12.57714  12.85714 12.26000   12.62286   352382100     11.016135
15   2009-01-23  12.40286  12.83857 12.35714   12.62286   190942500     11.016135
16   2009-01-26  12.69429  12.99571 12.61429   12.80571   173059600     11.175715
17   2009-01-27  12.88429  13.07857 12.82000   12.96143   154509600     11.311609
18   2009-01-28  13.16000  13.57143 13.07143   13.45714   215351500     11.744226
```



```
[ reached getOption("max.print") -- omitted 1515 entries ]
> # Sub-question 2-2:
> r.t <- as.data.frame(matrix(data = NA, nrow = (nrow(Apple)-1), ncol = 3))
> for (i in 2:nrow(Apple)) {
+   r.t[i-1, 1] <- rownames(Apple)[i]
+   r.t[i-1, 2] <- log(Apple$AAPL.Adjusted[i]) - log(Apple$AAPL.Adjusted[i-1])
+   r.t[i-1, 3] <- sum(r.t[1:(i-1), 2])
+ }
> names(r.t) <- c("Date", "Single L.r", "multiple time interval L.r")
> daily.log.return <- r.t[, 2]
> daily.log.return
   [1]  4.133749e-02 -1.663148e-02 -2.184519e-02  1.839909e-02 -2.313509e-02 -2.142463e-02 -1.077294e-02 -2.750988e-02 -2.311744e-02
  [10] -1.267291e-02 -5.146583e-02  5.752062e-02  6.462911e-02  0.000000e+00  1.438210e-02  1.208642e-02  3.753217e-02 -1.282060e-02
  [19] -3.134638e-02  1.519490e-02  1.593636e-02  6.111654e-03  3.063217e-02  3.323793e-02  2.759437e-02 -4.672906e-02 -1.037794e-02
  [28]  2.498994e-02 -1.108617e-03 -4.781767e-02 -1.693701e-03 -4.032791e-02  6.159451e-03 -4.772187e-02  3.725054e-02  1.003228e-02
  [37] -2.184715e-02  1.344466e-03 -1.545857e-02  4.877916e-03  3.119315e-02 -2.588864e-02 -4.066277e-02 -2.600937e-02  6.430531e-02
  [46]  4.468257e-02  3.883460e-02 -4.368737e-03 -5.330694e-03  4.347629e-02  1.849156e-02  9.843586e-04 -2.949295e-04  5.803270e-02
  [55] -1.083297e-02 -9.384589e-05  3.124656e-02 -2.787184e-02 -2.233445e-02  6.011090e-03  3.339704e-02  3.631859e-02  2.868594e-02
  [64]  2.098665e-02 -2.955860e-02  1.141318e-02  2.755680e-02  5.421246e-03 -1.601523e-02 -5.678836e-03  3.187338e-02  1.609023e-02
  [73] -2.394295e-02  1.040198e-02 -2.055374e-03  3.151192e-02 -1.203371e-02  6.676622e-03 -6.676622e-03  9.958135e-03  5.498706e-03
  [82]  1.114351e-02  3.725695e-02  4.833914e-03 -1.583567e-03 -2.630479e-02  1.006557e-03  2.937281e-03 -4.055874e-02 -4.042996e-02
  [91]  2.854497e-02 -4.319957e-03  3.396936e-02  6.297281e-03 -1.247457e-02 -1.351770e-02 -1.362115e-02  6.540580e-02  1.720826e-02
 [100]  1.506792e-02  5.463851e-03  2.573199e-02  1.004260e-03  1.041224e-02  1.960086e-02  6.449219e-03 -5.684691e-03 -7.886433e-03
 [109] -1.745778e-02 -2.141329e-03 -2.152338e-02 -6.445286e-03  1.908615e-03 -5.662915e-03  2.209976e-03  2.614885e-02 -1.524293e-02
 [118] -2.476387e-02  1.635704e-02  2.637070e-02  1.827910e-02 -3.304951e-03  3.234560e-03  2.804611e-03 -1.987018e-02 -1.012078e-02
 [127] -2.343055e-02  1.335168e-02 -6.286968e-03  1.571631e-02  2.720355e-02 -4.918375e-04  3.188927e-02  4.348164e-03  2.827039e-02
 [136]  7.615186e-03 -9.198008e-03  3.393668e-02  6.866957e-03  1.365606e-02  6.874538e-04 -6.247380e-04  1.874724e-04  1.709941e-02
 [145]  3.679054e-03  1.843521e-02 -5.301801e-03 -2.661430e-03 -7.294325e-03  9.714153e-03 -4.784459e-03 -1.154052e-02  1.511555e-02
```

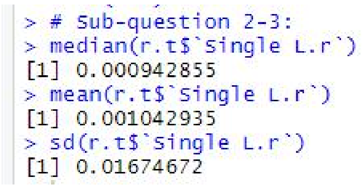
And here is the data frame contains daily return and multiple time interval return I get:



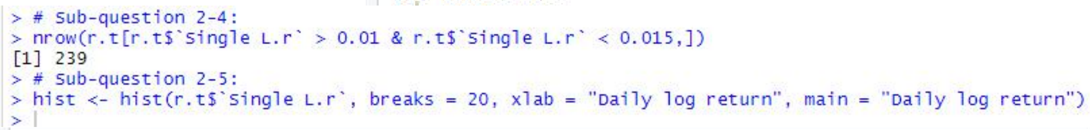
	Date	Single L.r	multiple time interval L.r
1	2009-01-05	4.133749e-02	0.0413374942
2	2009-01-06	-1.663148e-02	0.0247060126
3	2009-01-07	-2.184519e-02	0.0028608237
4	2009-01-08	1.839909e-02	0.0212599148
5	2009-01-09	-2.313509e-02	-0.0018751710
6	2009-01-12	-2.142463e-02	-0.0232998027
7	2009-01-13	-1.077294e-02	-0.0340727472
8	2009-01-14	-2.750988e-02	-0.0615826243
9	2009-01-15	-2.311744e-02	-0.0847000598
10	2009-01-16	-1.267291e-02	-0.0973729728
11	2009-01-20	-5.146583e-02	-0.1488388056
12	2009-01-21	5.752062e-02	-0.0913181864
13	2009-01-22	6.462911e-02	-0.0266890728
14	2009-01-23	0.000000e+00	-0.0266890728
15	2009-01-26	1.438210e-02	-0.0123069685
16	2009-01-27	1.208642e-02	-0.0002205455
17	2009-01-28	3.753217e-02	0.0373116265
18	2009-01-29	-1.282060e-02	0.0244910221
19	2009-01-30	-3.134638e-02	-0.0068553597
20	2009-02-02	1.519490e-02	0.0083395439
21	2009-02-03	1.593636e-02	0.0242758992
22	2009-02-04	6.111654e-03	0.0303875536
23	2009-02-05	3.063217e-02	0.0610197194
24	2009-02-06	3.323793e-02	0.0942576487
25	2009-02-09	2.759437e-02	0.1218520170
26	2009-02-10	-4.672906e-02	0.0751229593

Showing 1 to 31 of 2,515 entries, 3 total columns

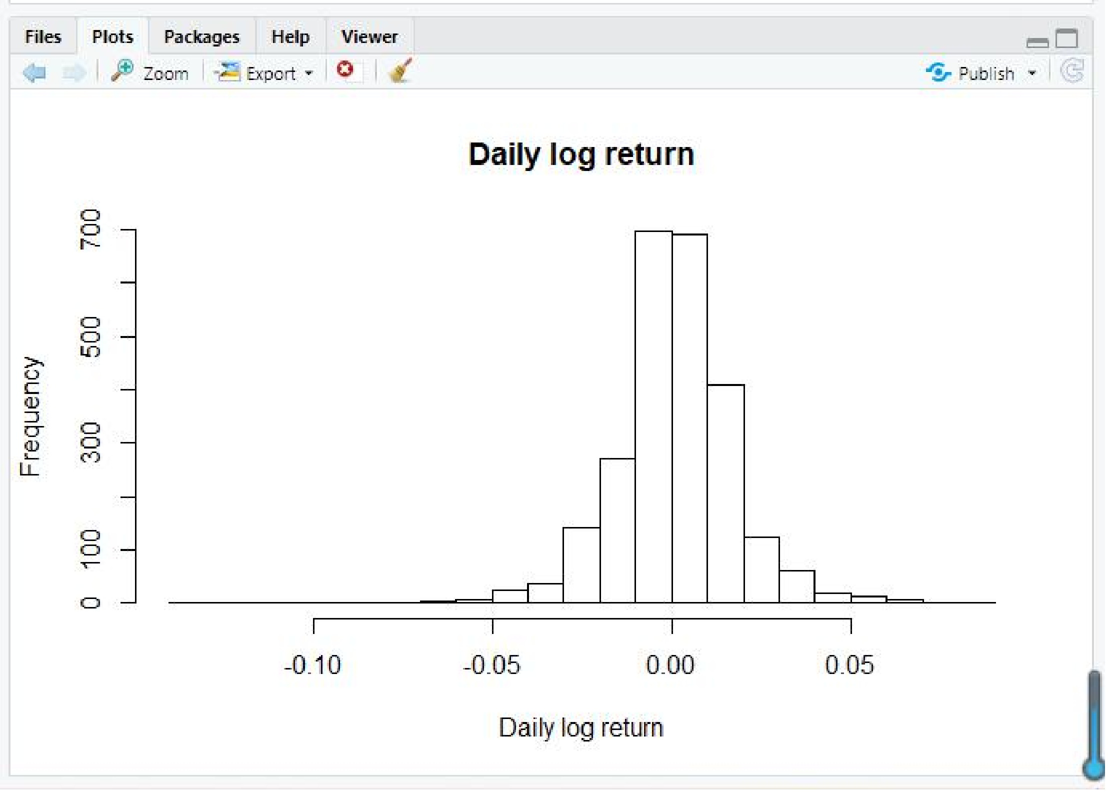
```
> View(r.t)
> 
```



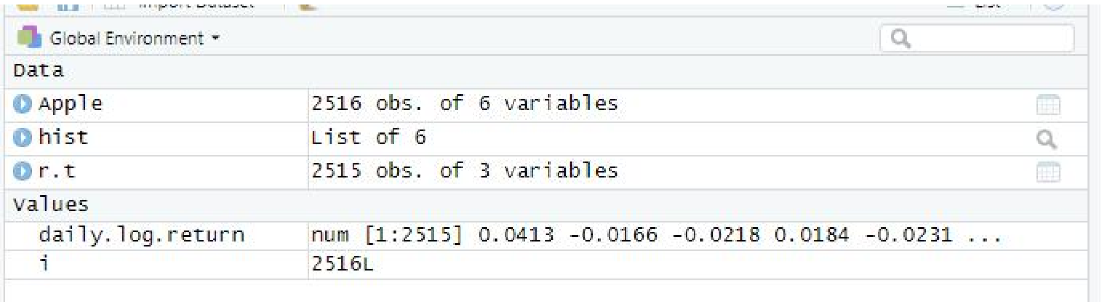
```
> # Sub-question 2-3:
> median(r.t$`Single L.r`)
[1] 0.000942855
> mean(r.t$`Single L.r`)
[1] 0.001042935
> sd(r.t$`Single L.r`)
[1] 0.01674672
```



```
> # Sub-question 2-4:
> nrow(r.t[r.t$`Single L.r` > 0.01 & r.t$`Single L.r` < 0.015,])
[1] 239
> # Sub-question 2-5:
> hist <- hist(r.t$`Single L.r`, breaks = 20, xlab = "Daily log return", main = "Daily log return")
> |
```



And here are new elements I get in this part:



```
Data
Apple              2516 obs. of 6 variables
hist               List of 6
r.t                2515 obs. of 3 variables
Values
daily.log.return   num [1:2515] 0.0413 -0.0166 -0.0218 0.0184 -0.0231 ...
i                  2516L
```