

## Report

# R Programming1

Mathematical Foundations of, Bioinformatics

Mahboobeh(Mariya) Golchinpour

## Problem 1: Multiplication table

Write a program in R to generate a PDF file for the standard 10 by 10 multiplication table.  
The first row and the first column take values 1 to 10.

## Solution

ابتدا دو بردار ۱۰ بعدی با اعداد ۱ تا ۱۰ تعریف کرده

سپس یک ماتریس ۱۰ در ۱۰ خالی به نام **result** تعریف میکنیم.

```
1 #+++++++create result matrix 10*10 multiplication table ++++++
2 # ++++++
3 v1<- c(1,2,3,4,5,6,7,8,9,10)
4 v2<- c(1,2,3,4,5,6,7,8,9,10)
5 result<- matrix(nrow = 10, ncol =10,dimnames = list(dim1 = v1, dim2 = v2)) # create empty matrix 10*10
6 print(result)
```

6:12 (Top Level) ↕

Console Terminal x Jobs x

R 4.1.2 · /media/mariya/Data/PHD/MFB/R\_Projects/HM1\_Programming/project/ ↗

```
> print(result)
      dim2
dim1 1  2  3  4  5  6  7  8  9 10
1    NA NA NA NA NA NA NA NA NA NA
2    NA NA NA NA NA NA NA NA NA NA
3    NA NA NA NA NA NA NA NA NA NA
4    NA NA NA NA NA NA NA NA NA NA
5    NA NA NA NA NA NA NA NA NA NA
6    NA NA NA NA NA NA NA NA NA NA
7    NA NA NA NA NA NA NA NA NA NA
8    NA NA NA NA NA NA NA NA NA NA
9    NA NA NA NA NA NA NA NA NA NA
10   NA NA NA NA NA NA NA NA NA NA
```

درایه های بردار اول را در تک تک درایه های بردار دوم ضرب کرده و حاصل ضرب را در **result** قرار میدهیم.

```
1.R* x
Source on Save
7 for (i in v1) {
8   for (j in v2) {
9     ij<- i*j
10    # print(ij)
11    result[i,j]<-ij
12  }
13 }
14 print(result)
12:4 (Top Level)

Console Terminal Jobs
R 4.1.2 · /media/mary/Data/PHD/MFB/R_Projects/HM1_Progr
> print(result)
      dim2
dim1  1  2  3  4  5  6  7  8  9 10
1    1  2  3  4  5  6  7  8  9 10
2    2  4  6  8 10 12 14 16 18 20
3    3  6  9 12 15 18 21 24 27 30
4    4  8 12 16 20 24 28 32 36 40
5    5 10 15 20 25 30 35 40 45 50
6    6 12 18 24 30 36 42 48 54 60
7    7 14 21 28 35 42 49 56 63 70
8    8 16 24 32 40 48 56 64 72 80
9    9 18 27 36 45 54 63 72 81 90
10   10 20 30 40 50 60 70 80 90 100
```

در ادامه result را در pdf ذخیره میکنیم.

The screenshot shows the R Studio interface. The R console on the left displays the same R code and output as the first block. The PDF viewer on the right shows a grid of numbers from 1 to 100, which is the result of the R code. The grid is 10 rows by 10 columns, with the first row containing numbers 1-10 and subsequent rows containing multiples of the row index.

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

## Problem 2: Prime number

A prime number (or a prime) is a natural number greater than 1 that is not a product of two smaller natural numbers [Wikipedia]. In this R exercise, your tasks are

(a) to determine whether an input natural number is a prime number

### Solution:

یک عدد به عنوان ورودی در نظر میگیریم و اول بودن و نبودن آن را چک میکنیم.

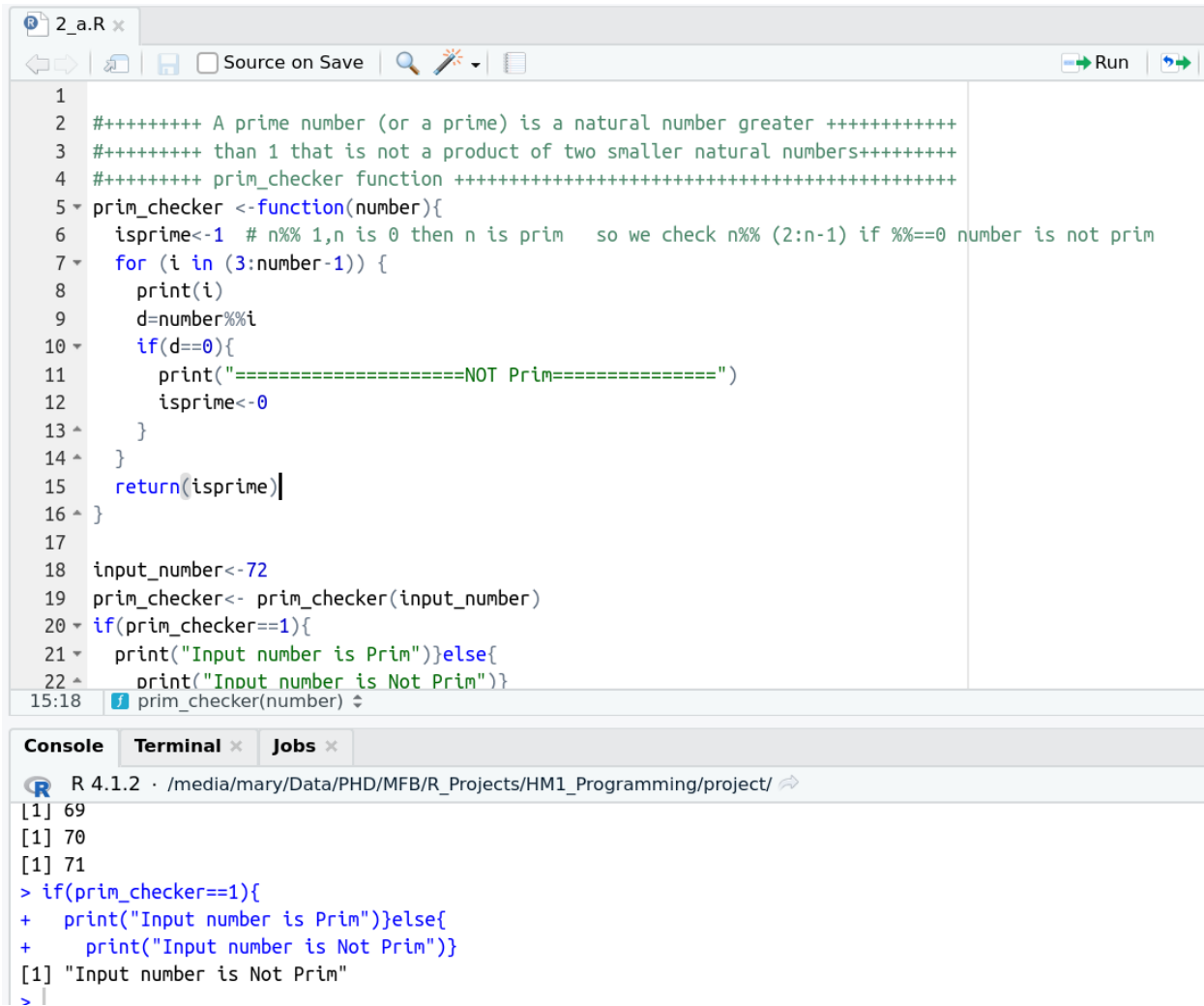
برای تشخیص اول بودن یک عدد با توجه به تعریف عدد اول که فقط بر خودش و یک بخش پذیر است. آن عدد را بر تمامی اعداد قبل خودش به جز خودش و یک تقسیم میکنیم.

اگر باقیمانده تقسیم صفر باشد آن عدد اول نخواهد بود.

```
2_a.R x
Source on Save
Run

1
2 ##### A prime number (or a prime) is a natural number greater #####
3 ##### than 1 that is not a product of two smaller natural numbers#####
4 ##### prim_checker function #####
5 prim_checker <-function(number){
6   isprime<-1 # n%% 1,n is 0 then n is prim so we check n%% (2:n-1) if %%==0 number is not prim
7   for (i in (3:number-1)) {
8     d=number%i
9     if(d==0){
10      print("=====NOT Prim=====")
11      isprime<-0
12    }
13  }
14  return(isprime)
15 }
16
17 input_number<-23
18 prim_checker<- prim_checker(input_number)
19 if(prim_checker==1){
20   print("Input number is Prim")}else{
21     print("Input number is Not Prim")}
22
17:17 (Top Level)

Console Terminal x Jobs x
R 4.1.2 · /media/mary/Data/PHD/MFB/R_Projects/HM1_Programming/project/
>
> input_number<-23
> prim_checker<- prim_checker(input_number)
> if(prim_checker==1){
+   print("Input number is Prim")}else{
+     print("Input number is Not Prim")}
[1] "Input number is Prim"
```



```
1
2 #+++++++ A prime number (or a prime) is a natural number greater ++++++++
3 #+++++++ than 1 that is not a product of two smaller natural numbers+++++++
4 #+++++++ prim_checker function ++++++++
5 prim_checker <-function(number){
6   isprime<-1 # n%% 1,n is 0 then n is prim so we check n%% (2:n-1) if %==0 number is not prim
7   for (i in (3:number-1)) {
8     print(i)
9     d=number%i
10    if(d==0){
11      print("=====NOT Prim=====")
12      isprime<-0
13    }
14  }
15  return(isprime)
16 }
17
18 input_number<-72
19 prim_checker<- prim_checker(input_number)
20 if(prim_checker==1){
21   print("Input number is Prim")}else{
22   print("Input number is Not Prim")}
15:18 prim_checker(number)
```

**Console** **Terminal** **Jobs**

R 4.1.2 · /media/mary/Data/PHD/MFB/R\_Projects/HM1\_Programming/project/

```
[1] 69
[1] 70
[1] 71
> if(prim_checker==1){
+   print("Input number is Prim")}else{
+     print("Input number is Not Prim")}
[1] "Input number is Not Prim"
>
```

**(b) to output the first 100000 prime numbers in a csv file**

در این قسمت برای اعداد ۲ تا ۱۰۰۰۰۰ تابع نوشته شده برای تشخیص عدد اول را فراخوانی کرده و اگر عدد اول باشد آن را در بردار **prim\_numbers** ذخیره میکنیم.

```
2_b.R x
Source on Save
7 d=number%%2
8 if(d==0){
9   print("=====NOT Prim=====")
10  isprime<-0
11 }
12 return(isprime)
13 }
14
15 prim_numbers<- c()
16 for (i in 2:100000) {
17   prim_check<- prim_checker(i)
18   print(i)
19   if(prim_check==1){
20     print("Input number is Prim")
21     prim_numbers <- append(prim_numbers, i)
22   }else{
23     print("Input number is Not Prim")}
24 }
25 write.csv(prim_numbers, file = "prim_numbers_100000.csv",row.names = FALSE)
26 # prim_numbers
25:7 (Top Level) ↕

Console Terminal x Jobs x
R 4.1.2 · /media/mary/Data/PHD/MFB/R_Projects/HM1_Programming/project/ ↗
[1] "=====NOT Prim=====
[1] "=====NOT Prim=====
[1] "=====NOT Prim=====
[1] "=====NOT Prim=====
[1] "=====NOT Prim=====
[1] 100000
[1] "Input number is Not Prim"
> write.csv(prim_numbers, file = "prim_numbers_100000.csv",row.names = FALSE)
> # prim_numbers
> |
```

در پایان بردار **prim\_numbers** را در فایل ذخیره می کنیم.

The screenshot displays the RStudio environment. The main editor window shows a text file with two columns of prime numbers, ranging from 9572 to 9593. The Environment pane on the right lists several objects: 'aframe' (7 obs. of 6 variables), 'data' (List of 1), 'df' (7 obs. of 4 variables), 'listt' (List of 4), 'ml' (num [1:4, 1:4] 0.), 'p' (num [1:8, 1:2] 0), and 'p1' (num [1:8, 1:2] 0). The Files pane shows a directory structure with files like '1.R', '2\_a.R', '2\_b.R', '3.R', '4.R', '7.R', 'dataframe.R', 'heatmap-bahonar.R', 'my\_boxplot.R', 'prim\_numbers\_100000.csv' (checked), 'project.Rproj', 'result.pdf', and 'segment.R'.

## Discuss the time complexity of your algorithms for both tasks

برای تشخیص اول بودن عدد  $N$  ان عدد را بر تمامی اعداد قبل خودش بجز خودش و یک تقسیم میکنیم در نتیجه  $N-2$  بار این کار انجام میشود. ( $N*(N-2)$ )

این الگوریتم دارای اردر  $O(n^2)$  است

### Problem 3: Birthday problem

Implement a function in R (from scratch) for the birthday problem and generate the Figure 1.5 of the textbook. Solve question 26-b of chapter 1 using your function.

$$p(\text{at least } k \text{ person match}) = 1 - p(\text{no match}) =$$

$$1 - (365/365) \cdot (364/365) \cdot (363/365) \cdot \dots \cdot (365 - k + 1)/365$$

$$p(1) = 1 - ((365 - 1 + 1)/365) = 0$$

$$p(3) = 1 - (365/365) \cdot (364/365) \cdot (363/365) = 0.0082$$

$$p(5) = 1 - (365/365) \cdot (364/365) \cdot (363/365) \cdot (362/365) \cdot (361/365) = 0.271$$

$$p(30) = 1 - (365/365) \cdot (364/365) \cdot (363/365) \cdot \dots \cdot (336/365) = 70.6$$

برای هر ۱۰۰ نفر احتمال مچ بودن را با استفاده از تابع **p\_match** محاسبه میکنیم. این تابع برای هر **n\_people** احتمال مچ بودن روز تولدشان را با پیاده سازی فورمول بالا حساب میکند سپس و احتمالش را در یک بردار **pall\_match** ذخیره میکنیم سپس با استفاده از **pall\_match**، پلات آن را رسم میکنیم.

Solve question 26-b of chapter 1 using your function.

# 26\_b) Find the probability that at least one person will get chosen more than once.

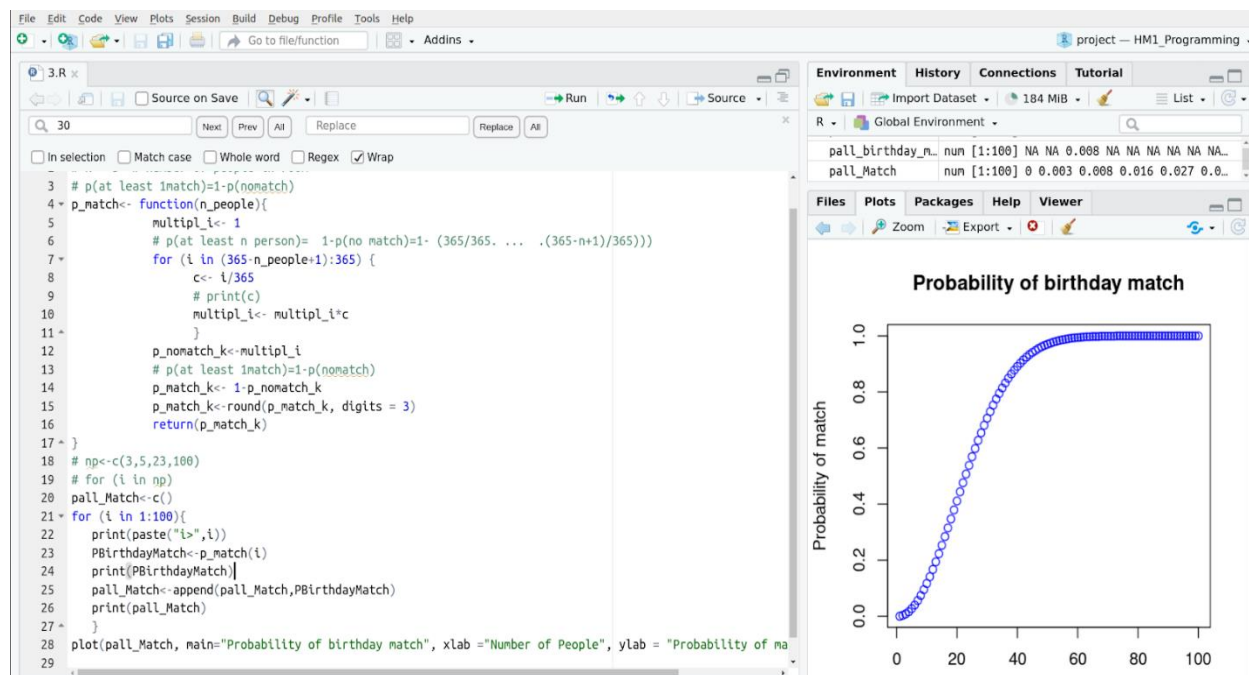
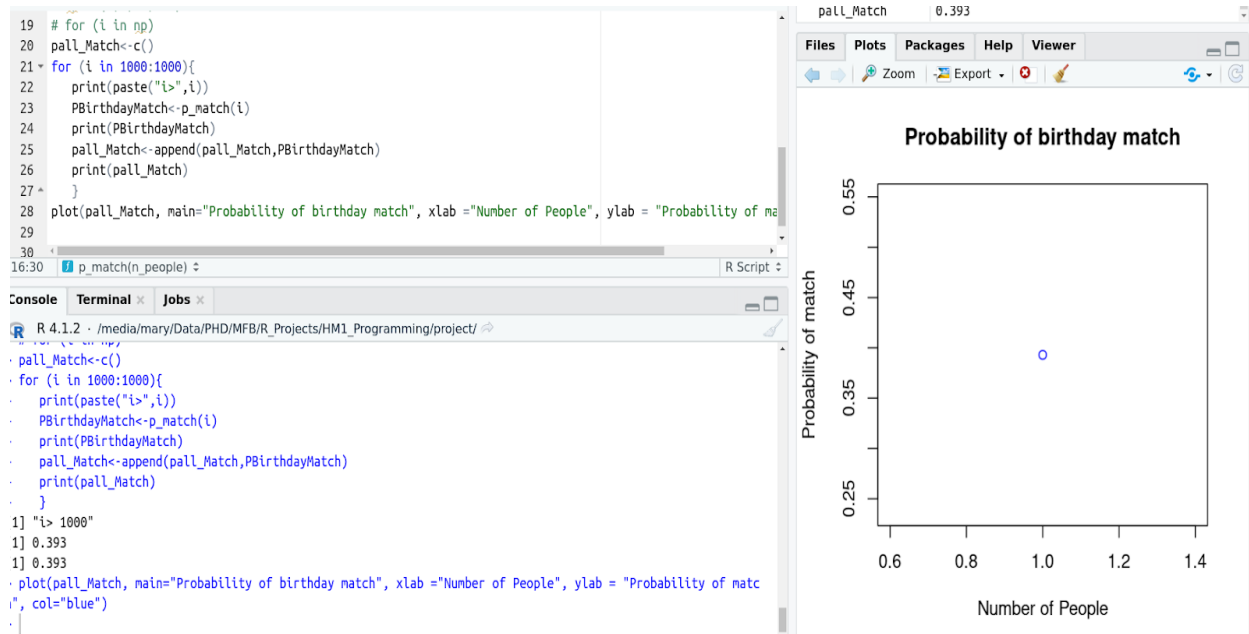
با توجه به فورمول تولد برای این سوال نیز از همان روش استفاده میکنیم. و تابع نوشته شده را برای ۱۰۰۰ نفر اجرا میکنیم. اگر ۱۰۰۰ نفر از ۱ میلیون نفر انتخاب شود احتمال اینکه حداقل یک نفر بیشتر از یک بار انتخاب شود با جایگذاری ۰.۳۹۳ است.

$$1 - (1000/1000) \cdot (365/365) \cdot (364/365) \cdot (363/365) \cdot \dots \cdot (365 - k + 1)/365$$

```
# p(at least 1match)=1-p(nomatch)
p_match<- function(n_people){
  multipl_i<- 1
  # p(at least n person)= 1-p(no match)=1-
  for (i in (1000000-n_people+1):1000000) {
    c<- i/1000000
    # print(c)
    multipl_i<- multipl_i*c
  }
  p_nomatch_k<-multipl_i
  # p(at least 1match)=1-p(nomatch)
  p_match_k<- 1-p_nomatch_k
  p_match_k<-round(p_match_k, digits = 3)
  return(p_match_k)
}
# np<-c(3,5,23,100)
# for (i in np)
pall_Match<-c()
```

**n\_people=1000**

**$p(n \text{ choose } k) = p(1000000 \text{ choose } 1000) = 1 - ((1000000 - n\_people + 1 / 1000000)) = 0.393$**

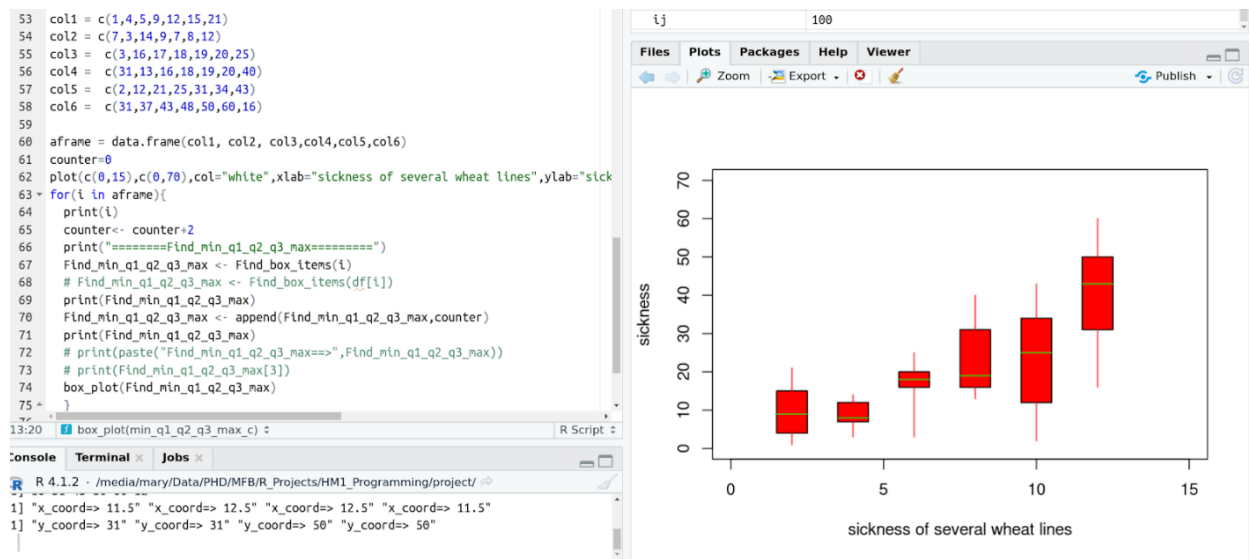




## Problem 4: Grouped and ordered boxplot

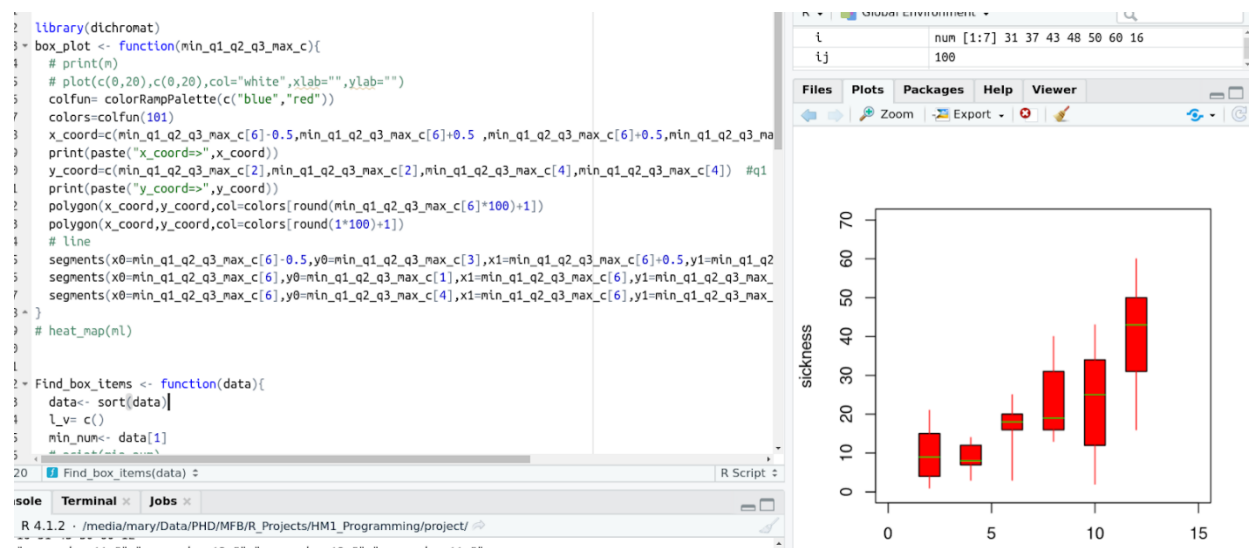
Implement a function in R to plot box plots without using any high-level built-in function in R.

Compare your plots with the code available in this link.



برای حل این سوال ابتدا ۶ بردار در نظر گرفته و در یک دیتا فریم ذخیره میکنیم تا برای هر کدام باکس پلات را رسم کنیم. سپس برای هر بردار مین- چارک اول- چارک دوم- چارک سوم- ماکسیمم را محاسبه و سپس یک شمارنده با فاصله دو برای هر بردار در نظر گرفتیم تا محور افقی باکس پلات را رسم کنیم ۱۲و۸و۴و۶و۳و۱۰و۱۲.

موارد محاسبه شده و شمارنده را در یک بردار ذخیره کرده و آن بردار را به تابعی که برای رسم باکس پلات نوشتیم به عنوان ورودی میدهیم.



از **segment** برای رسم خطوط از مین به چارک اول و از چارک سوم به ماکس و رسم خط چارک دوم استفاده کردم و از **polygon** برای رسم باکس استفاده کردم.