QnA: Attempting R Programming Sample Questions

John Karuitha; diakingathia005@gmail.com

12/16/22

Background

In this project, I attempt R programming sample questions available on this site https://www.w3resource.com/r-programming-exercises/. The questions range from basic to advanced R programming questions that could come handy in data analysis.

Question 1: Write a R program to take input from the user (name and age) and display the values. Also print the version of R installation.

The function readline() takes a prompt that a user can enter a value. Usually the value is a string. Thus if the input is a number, as.numeric function can do the conversion.

```
my_name <- readline(prompt = "Please enter your name: ")
```

```
Please enter your name:
```

```
my_age <- readline(prompt = "Please enter your age: ")
```

```
Please enter your age:
```

```
print(glue("Hello. My name is {my_name} and I am {my_age} years old."))
```

```
Hello. My name is  and I am  years old.
```

```r
## To print R version use R.Version()
R.Version()
```

```
$platform
[1] "x86_64-pc-linux-gnu"

$arch
[1] "x86_64"

$os
[1] "linux-gnu"

$system
[1] "x86_64, linux-gnu"

$status
[1] "Patched"

$major
[1] "4"

$minor
[1] "2.2"

$year
[1] "2022"

$month
[1] "11"

$day
[1] "10"

$`svn rev`
[1] "83330"

$language
[1] "R"

$version.string
[1] "R version 4.2.2 Patched (2022-11-10 r83330)"
```

```
$nickname
[1] "Innocent and Trusting"
```

Question 2: Write a R program to get the details of the objects in memory.

The ls() function, just like in linux, lists the objects in memory. on the other hand, sessionInfo() lists attached packages.

```
ls()
```

```
[1] "my_age"  "my_name"
```

```
sessionInfo()
```

```
R version 4.2.2 Patched (2022-11-10 r83330)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Zorin OS 16.2

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_GB.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_GB.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] glue_1.6.2      forcats_0.5.2   stringr_1.5.0   dplyr_1.0.10
 [5] purrr_0.3.5     readr_2.1.3     tidyr_1.2.1     tibble_3.1.8
 [9] ggplot2_3.4.0   tidyverse_1.3.2
```

```
loaded via a namespace (and not attached):
 [1] tidyselect_1.2.0    xfun_0.35           haven_2.5.1
 [4] gargle_1.2.1        colorspace_2.0-3    vctrs_0.5.1
 [7] generics_0.1.3      htmltools_0.5.4     yaml_2.3.6
[10] utf8_1.2.2          rlang_1.0.6         pillar_1.8.1
[13] withr_2.5.0         DBI_1.1.3           dbplyr_2.2.1
[16] modelr_0.1.10       readxl_1.4.1        lifecycle_1.0.3
[19] munsell_0.5.0       gtable_0.3.1        cellranger_1.1.0
[22] rvest_1.0.3         evaluate_0.19       knitr_1.41
[25] tzdb_0.3.0          fastmap_1.1.0       fansi_1.0.3
[28] broom_1.0.1         scales_1.2.1        backports_1.4.1
[31] googlesheets4_1.0.1 jsonlite_1.8.4      fs_1.5.2
[34] hms_1.1.2           digest_0.6.31       stringi_1.7.8
[37] grid_4.2.2          cli_3.4.1           tools_4.2.2
[40] magrittr_2.0.3      crayon_1.5.2        pkgconfig_2.0.3
[43] ellipsis_0.3.2      xml2_1.3.3          reprex_2.0.2
[46] googledrive_2.0.0   lubridate_1.9.0     timechange_0.1.1
[49] assertthat_0.2.1    rmarkdown_2.18      httr_1.4.4
[52] R6_2.5.1            compiler_4.2.2
```

Question 3: Write a R program to create a sequence of numbers from 20 to 50 and find the mean of numbers from 20 to 60 and sum of numbers from 51 to 91.

The : function is useful in this respect although the `seq` function is also a viable alternative.

```
my_seq <- 20:50

my_seq
```

```
 [1] 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
[26] 45 46 47 48 49 50
```

```
mean(20:60)
```

```
[1] 40
```

```
sum(51:91)
```

```
[1] 2911
```

Question 4: Write a R program to create a vector which contains 10 random integer values between -50 and +50.

The `sample` function takes the vector of numbers from which to pick a random sample and the size of the sample. The argument `replace` can be TRUE or FALSE depending on the goals of the sampling procedure.

```
sample(-50:50, size = 10, replace = TRUE)
```

```
 [1]   0  11  48  -5  22  43 -36 -17  -4  15
```

Question 5: Write a R program to get the first 10 Fibonacci numbers.

```
Fibonacci <- numeric(10)
Fibonacci[1] <- Fibonacci[2] <- 1
for (i in 3:10) Fibonacci[i] <- Fibonacci[i - 2] + Fibonacci[i - 1]
print("First 10 Fibonacci numbers:")
```

```
[1] "First 10 Fibonacci numbers:"
```

```
print(Fibonacci)
```

```
 [1]  1  1  2  3  5  8 13 21 34 55
```

Write a R program to print the numbers from 1 to 100 and print "Fizz" for multiples of 3, print "Buzz" for multiples of 5, and print "FizzBuzz" for multiples of both.

```
for (number in 1:100) {
    if (number %% 3 == 0 & number %% 5 == 0) {
        print(glue("{number} FizzBuzz"))
    } else if (number %% 3 == 0) {
        print(glue("{number} Fizz"))
    } else if (number %% 5 == 0) {
        print(glue("{number} buzz"))
    } else {
        print(glue("{number} Not divisible by 3, 5 or both"))
```

```
    }
}
```

```
1 Not divisible by 3, 5 or both
2 Not divisible by 3, 5 or both
3 Fizz
4 Not divisible by 3, 5 or both
5 buzz
6 Fizz
7 Not divisible by 3, 5 or both
8 Not divisible by 3, 5 or both
9 Fizz
10 buzz
11 Not divisible by 3, 5 or both
12 Fizz
13 Not divisible by 3, 5 or both
14 Not divisible by 3, 5 or both
15 FizzBuzz
16 Not divisible by 3, 5 or both
17 Not divisible by 3, 5 or both
18 Fizz
19 Not divisible by 3, 5 or both
20 buzz
21 Fizz
22 Not divisible by 3, 5 or both
23 Not divisible by 3, 5 or both
24 Fizz
25 buzz
26 Not divisible by 3, 5 or both
27 Fizz
28 Not divisible by 3, 5 or both
29 Not divisible by 3, 5 or both
30 FizzBuzz
31 Not divisible by 3, 5 or both
32 Not divisible by 3, 5 or both
33 Fizz
34 Not divisible by 3, 5 or both
35 buzz
36 Fizz
37 Not divisible by 3, 5 or both
38 Not divisible by 3, 5 or both
```

```
39 Fizz
40 buzz
41 Not divisible by 3, 5 or both
42 Fizz
43 Not divisible by 3, 5 or both
44 Not divisible by 3, 5 or both
45 FizzBuzz
46 Not divisible by 3, 5 or both
47 Not divisible by 3, 5 or both
48 Fizz
49 Not divisible by 3, 5 or both
50 buzz
51 Fizz
52 Not divisible by 3, 5 or both
53 Not divisible by 3, 5 or both
54 Fizz
55 buzz
56 Not divisible by 3, 5 or both
57 Fizz
58 Not divisible by 3, 5 or both
59 Not divisible by 3, 5 or both
60 FizzBuzz
61 Not divisible by 3, 5 or both
62 Not divisible by 3, 5 or both
63 Fizz
64 Not divisible by 3, 5 or both
65 buzz
66 Fizz
67 Not divisible by 3, 5 or both
68 Not divisible by 3, 5 or both
69 Fizz
70 buzz
71 Not divisible by 3, 5 or both
72 Fizz
73 Not divisible by 3, 5 or both
74 Not divisible by 3, 5 or both
75 FizzBuzz
76 Not divisible by 3, 5 or both
77 Not divisible by 3, 5 or both
78 Fizz
79 Not divisible by 3, 5 or both
80 buzz
81 Fizz
```

```
82 Not divisible by 3, 5 or both
83 Not divisible by 3, 5 or both
84 Fizz
85 buzz
86 Not divisible by 3, 5 or both
87 Fizz
88 Not divisible by 3, 5 or both
89 Not divisible by 3, 5 or both
90 FizzBuzz
91 Not divisible by 3, 5 or both
92 Not divisible by 3, 5 or both
93 Fizz
94 Not divisible by 3, 5 or both
95 buzz
96 Fizz
97 Not divisible by 3, 5 or both
98 Not divisible by 3, 5 or both
99 Fizz
100 buzz
```

Write a R program to extract first 10 english letter in lower case and last 10 letters in upper case and extract letters between 22nd to 24th letters in upper case.

```
letters[1:10]
```

```
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

```
LETTERS[-(1:10)]
```

```
 [1] "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
```

```
letters[22:24]
```

```
[1] "v" "w" "x"
```

```
LETTERS[22:24]
```

```
[1] "V" "W" "X"
```

Write a R program to find the factors of a given number.

```r
print_factors <- function(n) {
    print(paste("The factors of", n, "are:"))
    for (i in 1:n) {
        if ((n %% i) == 0) {
            print(i)
        }
    }
}

print_factors(13)
```

```
[1] "The factors of 13 are:"
[1] 1
[1] 13
```

Write a R program to find the maximum and the minimum value of a given vector.

```r
my_vector <- c(3, 4, 5, 6, 8)

for (number in my_vector) {
    max <- my_vector[1]
    if (number > max) {
        max <- number
    }
}

print(max)
```

```
[1] 8
```

```r
## or can use the bukt in max function
max(my_vector)
```

```
[1] 8
```

Write a R program to get the unique elements of a given string and unique numbers of vector.

```
another_vector <- c(4, 4, 5, 6)

unique(another_vector)
```

```
[1] 4 5 6
```

Write a R program to create three vectors a,b,c with 3 integers. Combine the three vectors to become a 3×3 matrix where each column represents a vector. Print the content of the matrix.

```
vector_1 <- c(3, 4, 5)
vector_2 <- c(5, 4, 7)
vector_3 <- c(1, 4, 0)

rbind(vector_1, vector_2, vector_3)
```

```
         [,1] [,2] [,3]
vector_1    3    4    5
vector_2    5    4    7
vector_3    1    4    0
```

Write a R program to create a list of random numbers in normal distribution and count occurrences of each value.

```
rnorm(100)
```

```
  [1] -1.29262194 -1.74183678  1.20055182 -0.26791961  0.30597661  0.18819518
  [7]  0.56034564  0.07560108 -1.69302604 -1.95834224 -1.21958271  0.80077236
 [13] -1.72528071 -1.66491566 -1.01692674  1.57035405 -0.04521971  2.27539303
 [19]  0.86209722 -1.48864288 -0.66794908  1.02142519 -1.27111581  0.53485146
 [25]  0.41378404  0.68121985  0.44173758  1.94116560 -0.64485032 -0.59799253
 [31] -0.12000218 -0.57842652  0.64533907 -1.72146742 -0.68054905 -0.88040679
 [37]  0.86657780 -1.11897344 -0.25308515 -0.12451859 -0.15589410  0.71189996
 [43] -1.34578330 -0.07098349 -2.31283763 -1.32770730  1.95866778  1.66976789
```

```
 [49] -0.59160855  0.11463701  0.24786642  0.46209553  0.41058152  0.92140514
 [55]  0.67792219 -1.06715670 -1.84858154  0.83898144  1.12520669 -1.79404112
 [61]  2.04619698 -0.61582320  0.06181184  0.24644992  1.79849266  1.44989741
 [67] -0.44612343 -0.75124866 -0.31020450  1.01218068  0.66347642 -0.05493652
 [73] -0.38918720  0.79670931 -0.63376976  0.88885213 -1.15402385 -0.03303657
 [79] -0.39113966 -0.46918509  2.19805057  0.72168765 -0.55157216 -1.20755199
 [85]  1.84838346 -1.64361536  1.14702641  0.22346860  1.36627327  0.15534885
 [91]  1.37244975 -1.02314519 -0.05810265 -0.15025800 -0.89193703  1.03975822
 [97] -0.35295851  0.55666960 -0.61690931  0.61849826
```

```
table(rnorm(100))
```

```
  -2.8779533287637   -2.48986320307903   -2.15539926146485   -1.87414393897969
                 1                   1                   1                   1
 -1.51045588833193   -1.49383712796095   -1.45973342635374   -1.43699573822069
                 1                   1                   1                   1
 -1.42218088228572   -1.37424309943358   -1.35067188544313   -1.34045010164227
                 1                   1                   1                   1
 -1.32363119382692   -1.32291811151721   -1.29865338503384   -1.22136020774063
                 1                   1                   1                   1
 -1.05350716800572   -1.03316374727785  -0.971825102587013  -0.917209802349558
                 1                   1                   1                   1
-0.884996687023808  -0.841847479890399  -0.828325547357757  -0.784304751062764
                 1                   1                   1                   1
 -0.70631917991472  -0.705071038906655  -0.670021098095324  -0.664266697957777
                 1                   1                   1                   1
-0.646178957166915  -0.641886937070352  -0.633130641594838  -0.580674626926787
                 1                   1                   1                   1
-0.537602121074141  -0.524610104970536  -0.456722581651079  -0.424812447644586
                 1                   1                   1                   1
-0.419451854819931  -0.396560974668212  -0.314647696113014   -0.31066450927893
                 1                   1                   1                   1
-0.296262216599071  -0.272998707859041  -0.271923551978691  -0.261331887887467
                 1                   1                   1                   1
 -0.25994646746309  -0.255518310433284  -0.220934402972492   -0.21565523959393
                 1                   1                   1                   1
 -0.173956784427216  -0.105231270835073 -0.0818653204425048 -0.0766545015715369
                 1                   1                   1                   1
-0.0585415348187799 -0.0116946351771151   0.101250210877125   0.149403337379737
                 1                   1                   1                   1
```

```
 0.215209273169607   0.256565700168459   0.262069039323194   0.352996871914942
                 1                   1                   1                   1
 0.374796333280823   0.414899018863463   0.432197880829738   0.434899993313107
                 1                   1                   1                   1
 0.483486883751105   0.536557835575211   0.545338444744307    0.58342199989157
                 1                   1                   1                   1
 0.600383810045686   0.639937586904964   0.654995277535123   0.660133389436089
                 1                   1                   1                   1
  0.71454869094345   0.775384191144542   0.798165300879335   0.811603153417349
                 1                   1                   1                   1
 0.824721264287784   0.906211640271205   0.998678766823776    1.00778836535572
                 1                   1                   1                   1
  1.07833645100385    1.14977007999228    1.15634472122049    1.26340811353609
                 1                   1                   1                   1
  1.32833863542241    1.33841614581942    1.34629913642959    1.35237289861166
                 1                   1                   1                   1
  1.37309545706638    1.41113239548546    1.43656390292161    1.51136762209508
                 1                   1                   1                   1
  1.55658290566271    1.66028720789287    1.77581182299159    1.89617762368669
                 1                   1                   1                   1
  1.99130039163503    2.01779946923295    2.37117506167283    2.53117236112526
                 1                   1                   1                   1
```

Write a R program to read the .csv file and display the content.

Here I use a sample of csv files available on this site https://people.sc.fsu.edu/~jburkardt/data/csv/csv.html

```
my_data <- curl::curl("https://people.sc.fsu.edu/~jburkardt/data/csv/airtravel.csv")

my_csv <- read.csv(my_data)

my_csv
```

```
   Month X1958 X1959 X1960
1    JAN   340   360   417
2    FEB   318   342   391
3    MAR   362   406   419
4    APR   348   396   461
5    MAY   363   420   472
6    JUN   435   472   535
```

```
7    JUL   491   548   622
8    AUG   505   559   606
9    SEP   404   463   508
10   OCT   359   407   461
11   NOV   310   362   390
12   DEC   337   405   432
```

```r
my_csv %>%
    pivot_longer(-Month,
        names_to = "year",
        values_to = "passengers"
    ) %>%
    relocate(year) %>%
    mutate(year = str_remove_all(year, "X")) %>%
    arrange(year)
```

```
# A tibble: 36 x 3
   year  Month passengers
   <chr> <chr>      <int>
 1 1958  JAN          340
 2 1958  FEB          318
 3 1958  MAR          362
 4 1958  APR          348
 5 1958  MAY          363
 6 1958  JUN          435
 7 1958  JUL          491
 8 1958  AUG          505
 9 1958  SEP          404
10 1958  OCT          359
# ... with 26 more rows
```

Write a R program to create three vectors numeric data, character data and logical data. Display the content of the vectors and their type.

```r
double_vector <- c(1, 4, 5)
logical_vector <- c(TRUE, FALSE)
string_vector <- c("Njogu", "paul")

class(double_vector)
```

```
[1] "numeric"
```

```
class(logical_vector)
```

```
[1] "logical"
```

```
class(string_vector)
```

```
[1] "character"
```

Write a R program to create a 5 x 4 matrix , 3 x 3 matrix with labels and fill the matrix by rows and 2 × 2 matrix with labels and fill the matrix by columns.

```
five_by_four <- 1:20

matrix(five_by_four,
    nrow = 5, byrow = TRUE,
    dimnames = list(c("One", "Two", "Three", "Four", "Five"), c("One", "Two", "Three", "Fo
)
```

```
      One Two Three Four
One     1   2     3    4
Two     5   6     7    8
Three   9  10    11   12
Four   13  14    15   16
Five   17  18    19   20
```

```
three_by_three <- 1:9

matrix(three_by_three, nrow = 3, byrow = TRUE, dimnames = list(c("One", "Two", "Three"), c
```

```
      One Two Three
One     1   2     3
Two     4   5     6
Three   7   8     9
```

```r
two_by_two <- 1:4

matrix(two_by_two,
    nrow = 2, byrow = FALSE,
    dimnames = list(c("One", "Two"), c("One", "Two"))
)
```

```
    One Two
One   1   3
Two   2   4
```

Write a R program to create an array, passing in a vector of values and a vector of dimensions. Also provide names for each dimension.

```r
a <- array(
    6:30,
    dim = c(4, 3, 2),
    dimnames = list(
        c("Col1", "Col2", "Col3", "Col4"),
        c("Row1", "Row2", "Row3"),
        c("Part1", "Part2")
    )
)
print(a)
```

```
, , Part1

     Row1 Row2 Row3
Col1    6   10   14
Col2    7   11   15
Col3    8   12   16
Col4    9   13   17

, , Part2

     Row1 Row2 Row3
Col1   18   22   26
Col2   19   23   27
Col3   20   24   28
Col4   21   25   29
```

Write a R program to create an array with three columns, three rows, and two "tables", taking two vectors as input to the array. Print the array.

```r
v1 <- c(1, 3, 5, 7)
v2 <- c(2, 4, 6, 8, 10)
arra1 <- array(c(v1, v2), dim = c(3, 3, 2))
print(arra1)
```

```
, , 1

     [,1] [,2] [,3]
[1,]    1    7    6
[2,]    3    2    8
[3,]    5    4   10

, , 2

     [,1] [,2] [,3]
[1,]    1    7    6
[2,]    3    2    8
[3,]    5    4   10
```

Write a R program to create a list of elements using vectors, matrices and a functions. Print the content of the list.

```r
list(c(1, 3), matrix(1:4, nrow = 2), mean)
```

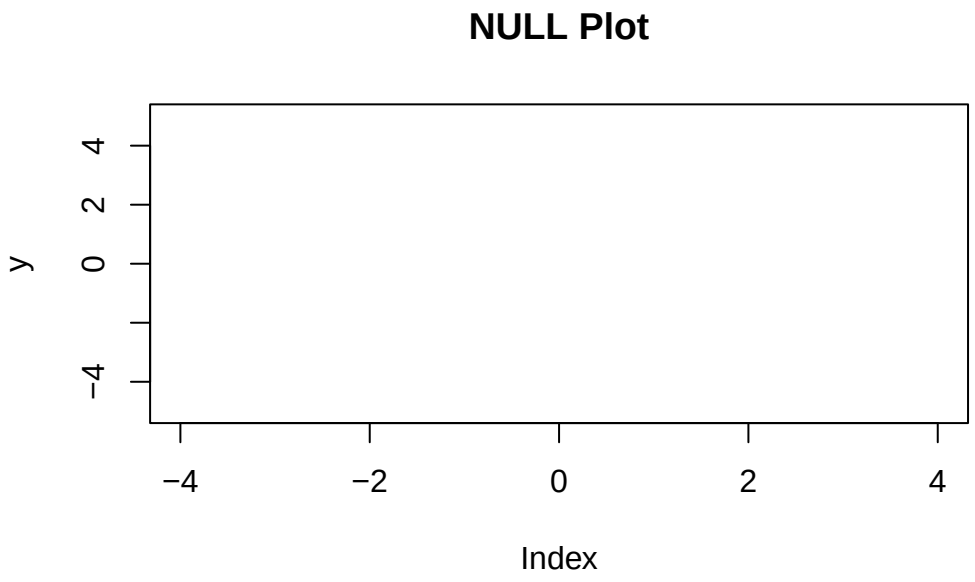
```
[[1]]
[1] 1 3

[[2]]
     [,1] [,2]
[1,]    1    3
[2,]    2    4

[[3]]
function (x, ...)
UseMethod("mean")
<bytecode: 0x562889553ad0>
<environment: namespace:base>
```

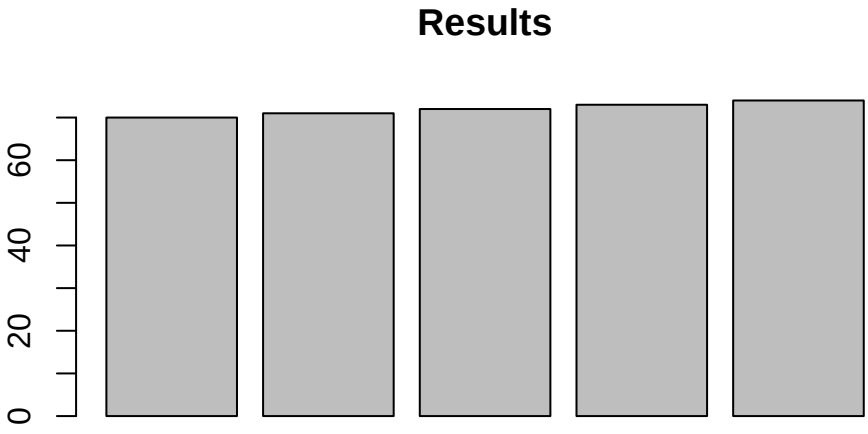
Write a R program to draw an empty plot and an empty plot specify the axes limits of the graphic.

```r
x <- NULL
y <- NULL
plot(x, y, xlim = c(-4, 4), ylim = c(-5, 5), main = "NULL Plot")
```



Write a R program to create a simple bar plot of five subjects marks.

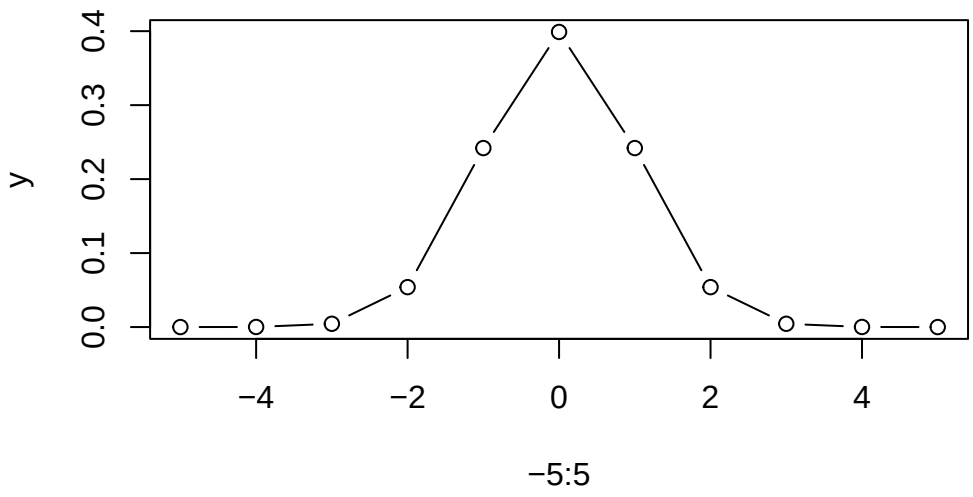
```r
marks <- c(70:74)
barplot(marks, main = "Results")
```



Write a R program to create bell curve of a random normal distribution.

```
y <- dnorm(x = seq(from = -5, to = 5, by = 1))

plot(-5:5, y, type = "both")
```



Write a R program to compute sum, mean and product of a given vector elements.

```
sum(marks)
```

```
[1] 360
```

```
mean(marks)
```

```
[1] 72
```

```
prod(marks)
```

```
[1] 1933051680
```

Write a R program to create a list of heterogeneous data, which include character, numeric and logical vectors. Print the lists.

```
list(names = c("Jane", "Karuitha"), numerals = 1:4, logic = c(TRUE, FALSE))
```

```
$names
[1] "Jane"     "Karuitha"

$numerals
[1] 1 2 3 4

$logic
[1]  TRUE FALSE
```

Write a R program to create a Dataframes which contain details of 5 employees and display the details.

```
my_data <- data.frame(names = c("Jane", "Karuitha"), age = c(19, 33), subject = c("Math",

summary(my_data)
```

```
   names                age          subject
 Length:2           Min.   :19.0   Length:2
 Class :character   1st Qu.:22.5   Class :character
 Mode  :character   Median :26.0   Mode  :character
                    Mean   :26.0
                    3rd Qu.:29.5
                    Max.   :33.0
```

Write a R program to create the system's idea of the current date with and without time.

```
Sys.Date()
```

```
[1] "2022-12-16"
```

```r
Sys.time()
```

```
[1] "2022-12-16 00:46:15 EAT"
```

map function applies a function to many elements

```r
my_list = list(height = c(1.7, 2), age = c(45, 30), weight = c(84, 70))
purrr::map_dbl(my_list, mean)
```

```
height    age weight
  1.85  37.50  77.00
```

```r
my_data %>% skimr::skim_without_charts()
```

Table 1: Data summary

Name	Piped data
Number of rows	2
Number of columns	3
_____	
Column type frequency:	
character	2
numeric	1
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
names	0	1	4	8	0	2	0
subject	0	1	4	7	0	2	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
age	0	1	26	9.9	19	22.5	26	29.5	33