DAY 2 ASSESSMENT

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.

CODE:

> name <- readline(prompt = "Enter your name: ")

Enter your name: INDHU

>

> age <- readline(prompt = "Enter your age: ")

Enter your age:

>

> 6

[1] 6

> cat("Your name is", name, "and your age is", age, "\n")

Your name is INDHU and your age is

> cat("R version", paste(R.version$major, R.version$minor, sep = "."), "\n")

R version 4.2.3

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

CODE:

> a <- 1:100

> b <- "INDHU"

> c <- data.frame(x = 1:10, y = 11:20)

> objects <- ls()

> for (obj in objects) {

+ size <- object.size(get(obj))

+ cat(paste("Object name:", obj, " Size:", size, "\n"))

+ }

Object name: a Size: 448

Object name: b Size: 112

Object name: c Size: 944

Object name: obj Size: 112

Object name: objects Size: 432

Object name: size Size: 288

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.

CODE:

> seq\_numbers =20:50

> mean\_numbers = mean(20:60)

> sum\_numbers = sum(51:91)

> cat("Sequence of numbers from 20 to 50: ", seq\_numbers, "\n")

Sequence of numbers from 20 to 50: 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 45 46 47 48 49 50

> cat("Mean of numbers from 20 to 60: ", mean\_numbers, "\n")

Mean of numbers from 20 to 60: 40

> cat("Sum of numbers from 51 to 91: ", sum\_numbers, "\n")

Sum of numbers from 51 to 91: 2911

4. Write a R program to create a vector which contains 10 random integer values

between -50 and +50.

CODE:

> set.seed(123)

> my\_vector <- sample(-50:50, 10, replace = TRUE)

> print(my\_vector)

[1] -20 28 0 -37 16 -9 -1 -8 50 -37

5. Write a R program to get all prime numbers up to a given number (based on

the sieve of Eratosthenes).

CODE:

> prime\_numbers <- function(n) {

+ if (n >= 2) {

+ x = seq(2, n)

+ prime\_nums = c()

+ for (i in seq(2, n)) {

+ if (any(x == i)) {

+ prime\_nums = c(prime\_nums, i)

+ x = c(x[(x %% i) != 0], i)

+ }

+ }

+ return(prime\_nums)

+ }

+ else

+ {

+ stop("Input number should be at least 2.")

+ }

+ }

> prime\_numbers(12)

[1] 2 3 5 7 11

6. Write a R program to extract first 10 english letter in lower case and last 10

letters in upper case and extract letters between 22 nd  to 24 th  letters in upper case.

CODE:

> first\_10\_lower = letters[1:10]

> print(first\_10\_lower)

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

> last\_10\_upper <- tail(LETTERS, 10)

> print(last\_10\_upper)

[1] "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"

>

> mid\_upper <- LETTERS[22:24]

> print(mid\_upper)

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

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

vector.

CODE:

> my\_vector <- c(12, 34, 56, -7, 8, 90)

> max\_value= max(my\_vector)

> print(max\_value)

[1] 90

> min\_value =min(my\_vector)

> print(min\_value)

[1] -7

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

numbers of vector.

CODE:

> my\_string <- "hello world"

> my\_vector <- c(1, 2, 1, 3, 4, 2, 5, 4, 6)

> # get unique elements of the string

> unique\_string <- unique(strsplit(my\_string, "")[[1]])

> # get unique elements of the vector

> unique\_vector <- unique(my\_vector)

> # print the results

> cat("Unique elements of the string:", unique\_string, "\n")

Unique elements of the string: h e l o w r d

> cat("Unique elements of the vector:", unique\_vector, "\n")

Unique elements of the vector: 1 2 3 4 5 6

9. 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.

CODE:

> a <- c(1, 2, 3)

> b <- c(4, 5, 6)

> c <- c(7, 8, 9)

> # combine the vectors into a matrix

> my\_matrix <- cbind(a, b, c)

> # print the content of the matrix

> print(my\_matrix)

a b c

[1,] 1 4 7

[2,] 2 5 8

[3,] 3 6 9

10. Write a R program to create a list of random numbers in normal distribution

and count occurrences of each value.

CODE:

> set.seed(123)

> # generate a list of random numbers in normal distribution

> my\_list <- rnorm(100, mean = 0, sd = 1)

> # count the occurrences of each value

> my\_counts <- table(my\_list)

> # print the results

> print(my\_counts)

my\_list

-2.30916887564081 -1.96661715662964 -1.68669331074241 -1.54875280423022 -1.26539635156826

1 1 1 1 1

-1.26506123460653 -1.22071771225454 -1.13813693701195 -1.12310858320335 -1.07179122647558

1 1 1 1 1

-1.06782370598685 -1.02642090030678 -1.02600444830724 -1.01857538310709 -0.72889122929114

1 1 1 1 1

-0.709200762582393 -0.694706978920513 -0.688008616467358 -0.686852851893526 -0.627906076039371

1 1 1 1 1

-0.625039267849257 -0.600259587147127 -0.560475646552213 -0.555841134754075 -0.502323453109302

1 1 1 1 1

-0.491031166056535 -0.472791407727934 -0.466655353623219 -0.445661970099958 -0.402884835299076

1 1 1 1 1

-0.380471001012383 -0.370660031792409 -0.33320738366942 -0.325931585531227 -0.305962663739917

1 1 1 1 1

-0.295071482992271 -0.284773007051009 -0.235700359100477 -0.23017748948328 -0.225770985659268

1 1 1 1 1

-0.220486561818751 -0.217974914658295 -0.207917278019599 -0.138891362439045 -0.0833690664718293

1 1 1 1 1

-0.0619117105767217 -0.0428704572913161 -0.028546755348703 0.00576418589988693 0.0530042267305041

1 1 1 1 1

0.070508391424576 0.11068271594512 0.123854243844614 0.129287735160946 0.153373117836515

1 1 1 1 1

0.18130347974915 0.215941568743973 0.238731735111441 0.253318513994755 0.303528641404258

1 1 1 1 1

0.331781963915697 0.359813827057364 0.379639482759882 0.38528040112633 0.400771450594052

1 1 1 1 1

0.426464221476814 0.435181490833803 0.448209778629426 0.460916205989202 0.497850478229239

1 1 1 1 1

0.54839695950807 0.553917653537589 0.584613749636069 0.644376548518833 0.688640254100091

1 1 1 1 1

0.701355901563686 0.779965118336318 0.821581081637487 0.837787044494525 0.878133487533042

1 1 1 1 1

0.895125661045022 0.922267467879738 0.993503855962119 1.00573852446226 1.0255713696967

1 1 1 1 1

1.09683901314935 1.14880761845109 1.20796199830499 1.22408179743946 1.25381492106993

1 1 1 1 1

1.36065244853001 1.36860228401446 1.51647060442954 1.53261062618519 1.55870831414912

1 1 1 1 1

1.71506498688328 1.78691313680308 2.05008468562714 2.16895596533851 2.18733299301658

1 1 1 1 1

11. Write a R program to create three vectors numeric data, character data and

logical data. Display the content of the vectors and their type.

CODE:

> numeric\_vec <- c(1.2, 2.3, 3.4, 4.5)

> character\_vec <- c("apple", "banana", "orange", "pear")

> logical\_vec <- c(TRUE, FALSE, TRUE, FALSE)

> # display the content of the vectors

> print(numeric\_vec)

[1] 1.2 2.3 3.4 4.5

> print(character\_vec)

[1] "apple" "banana" "orange" "pear"

> print(logical\_vec)

[1] TRUE FALSE TRUE FALSE

> # display the type of the vectors

> print(typeof(numeric\_vec))

[1] "double"

> print(typeof(character\_vec))

[1] "character"

> print(typeof(logical\_vec))

[1] "logical"

12. 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.

CODE:

> matrix1 <- matrix(1:20, nrow = 5, ncol = 4, byrow = TRUE)

> print(matrix1)

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

[1,] 1 2 3 4

[2,] 5 6 7 8

[3,] 9 10 11 12

[4,] 13 14 15 16

[5,] 17 18 19 20

> # create a 3x3 matrix with labels and fill by rows

> matrix2 <- matrix(c("A", "B", "C", "D", "E", "F", "G", "H", "I"), nrow = 3, ncol = 3, byrow = TRUE, dimnames = list(c("Row1", "Row2", "Row3"), c("Col1", "Col2", "Col3")))

> print(matrix2)

Col1 Col2 Col3

Row1 "A" "B" "C"

Row2 "D" "E" "F"

Row3 "G" "H" "I"

> # create a 2x2 matrix with labels and fill by columns

> matrix3 <- matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = FALSE, dimnames = list(c("Row1", "Row2"), c("Col1", "Col2")))

> print(matrix3)

Col1 Col2

Row1 1 3

Row2 2 4

13. 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.

CODE:

> values <- c(1, 2, 3, 4, 5, 6)

> # create a vector of dimensions

> dims <- c(2, 3)

> # create the array with dimensions and names

> my\_array <- array(values, dim = dims, dimnames = list(c("Row1", "Row2"), c("Col1", "Col2", "Col3")))

> # print the array

> print(my\_array)

Col1 Col2 Col3

Row1 1 3 5

Row2 2 4 6

14. Write a R program to create an array with three columns, three rows, and two

&quot;tables&quot;, taking two  vectors as input to the array.  Print the array.

CODE:

> vector1 <- 1:18

> vector2 <- 19:36

> # create a 3 x 3 x 2 array with the two vectors as input

> my\_array <- array(c(vector1, vector2), dim = c(3, 3, 2))

> # print the array

> print(my\_array)

, , 1

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

[1,] 1 4 7

[2,] 2 5 8

[3,] 3 6 9

, , 2

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

[1,] 10 13 16

[2,] 11 14 17

[3,] 12 15 18

15. Write a R program to create a list of elements using vectors, matrices and a

functions. Print the content of the list.

CODE:

> my\_vector <- c(1, 2, 3)

> my\_matrix <- matrix(1:6, nrow = 2)

> my\_function <- function(x) {x^2}

> # create a list with the vector, matrix, and function as elements

> my\_list <- list(my\_vector, my\_matrix, my\_function)

> # print the contents of the list

> print(my\_list)

[[1]]

[1] 1 2 3

[[2]]

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

[1,] 1 3 5

[2,] 2 4 6

[[3]]

function(x) {x^2}