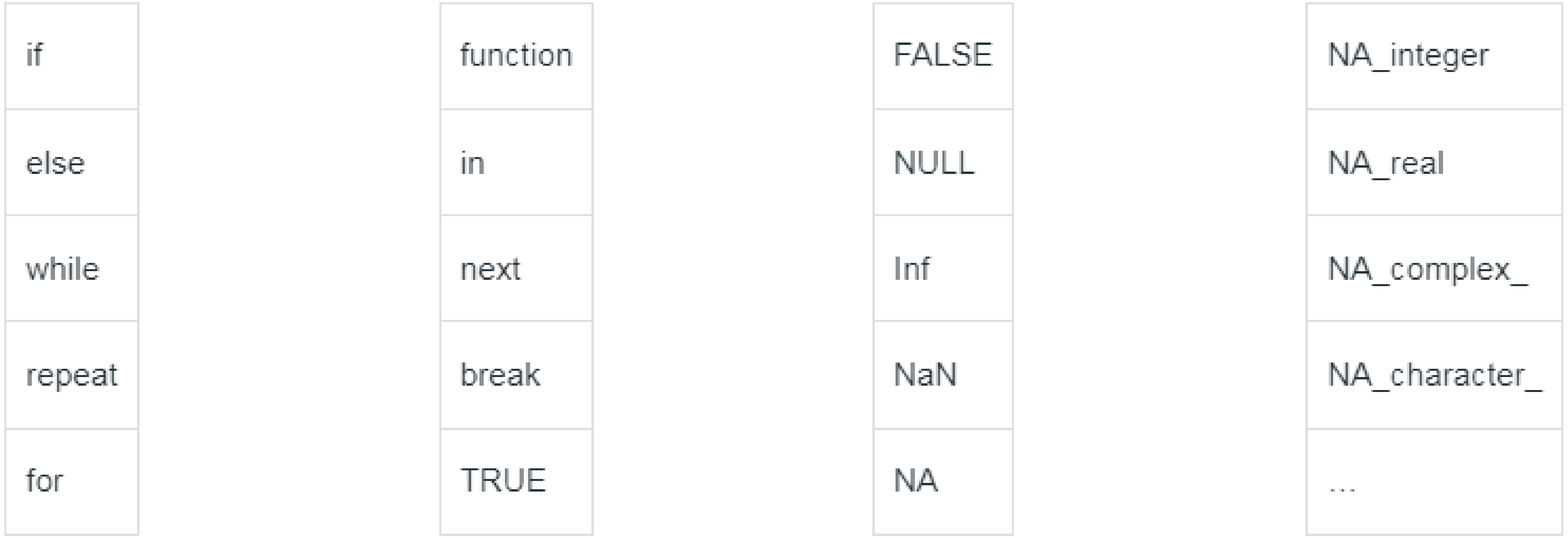
Data analysis using R
Unit 1 –Lec 3 & 4

By

Dr.Parkavi.A

R – Keywords

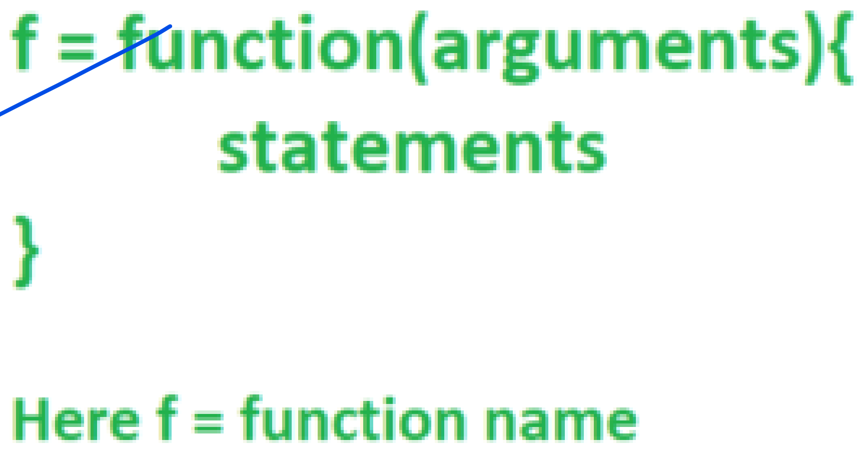
- Keywords are specific reserved words in R



if	function	FALSE	NA_integer
else	in	NULL	NA_real
while	next	Inf	NA_complex_
repeat	break	NaN	NA_character_
for	TRUE	NA	...

Functions in R Programming

- A function accepts input arguments and produces the output by executing valid R commands that are inside the function
- Functions are useful when you want to perform a certain task multiple times.
- **Creating a Function in R Programming**



```
f = function(arguments){
        statements
}
```

Here f = function name

Parameters or Arguments in R Functions

```
# function to add 2 numbers
add_num <- function(a,b)
{
sum_result <- a+b
return(sum_result)
}

# calling add_num function
sum = add_num(35,34)

#printing result
print(sum)
```

Functions contd..

- No. of Parameters
- Default Value of Parameter
- Return Value
- Calling a Function in R

Passing Arguments to Functions in R Programming Language

- **Case 1**: Generally in R, the **arguments are passed** to the function in the **same order** as in the function definition.
- **Case 2**: If you **do not want to follow any order** what you can do is you can pass the arguments using the names of the arguments in any order.
- **Case 3**: If the arguments are not passed the **default values** are used to execute the function.

Eg

```
# A simple R program to demonstrate
# passing arguments to a function
Rectangle = function(length=5, width=4){
area = length * width
return(area)
}
# Case 1:
print(Rectangle(2, 3))
# Case 2:
print(Rectangle(width = 8, length = 4))
# Case 3:
print(Rectangle())
```

Types of Function in R Language

- **Built-in Function:** Built-in functions in R are pre-defined functions that are available in R programming languages to perform common tasks or operations.
- **User-defined Function:** R language allow us to write our own function.

Built-in Function in R Programming Language

Eg:

```
# Find sum of numbers 4 to 6.
print(sum(4:6))

# Find max of numbers 4 and 6.
print(max(4:6))

# Find min of numbers 4 and 6.
print(min(4:6))
```

Functions list

Functions	Syntax
Mathematical Functions	
abs()	calculates a number’s absolute value.
sqrt()	calculates a number’s square root.
round()	rounds a number to the nearest integer.
exp()	calculates a number’s exponential value
log()	which calculates a number’s natural logarithm.
cos(), sin(), and tan()	calculates a number’s cosine, sine, and tang.

Functions list

Statistical Functions	
mean()	A vector’s arithmetic mean is determined by the mean() function.
median()	A vector’s median value is determined by the median() function.
cor()	calculates the correlation between two vectors.
var()	calculates the variance of a vector and calculates the standard deviation of a vector.

Contd..

Data Manipulation Functions	
unique()	returns the unique values in a vector.
subset()	subsets a data frame based on conditions.
aggregate()	groups data according to a grouping variable.
order()	uses ascending or descending order to sort a vector.

Contd…

File Input/Output Functions	
read.csv()	reads information from a CSV file.
Write.csv()	publishes information to write a CSV file.
Read. table()	reads information from a tabular.
Write.table()	creates a tabular file with data.

User-defined Functions in R Programming Language

```
# A simple R function to check
# whether x is even or odd

evenOdd = function(x){
if(x %% 2 == 0)
        return("even")
else
        return("odd")
}

print(evenOdd(4))
print(evenOdd(3))
```

Eg. Single Input Single Output

```
# A simple R function to calculate
# area of a circle

areaOfCircle = function(radius){
area = pi*radius^2
return(area)
}

print(areaOfCircle(2))
```

Eg. Multiple Input Multiple Output

```
# A simple R function to calculate
# area and perimeter of a rectangle
Rectangle = function(length, width){
area = length * width
perimeter = 2 * (length + width)
# create an object called result which is
# a list of area and perimeter
result = list("Area" = area, "Perimeter" = perimeter)
return(result)
}
resultList = Rectangle(2, 3)
print(resultList["Area"])
print(resultList["Perimeter"])
```

Eg. Inline Functions in R Programming Language

```
# A simple R program to
# demonstrate the inline function

f = function(x) x^2*4+x/3

print(f(4))
print(f(-2))
print(f(0))
```

Eg. Lazy Evaluations of Functions in R Programming Language

```
# A simple R program to demonstrate
# Lazy evaluations of functions

Cylinder = function(diameter, length, radius ){
volume = pi*diameter^2*length/4
return(volume)
}

# This'll execute because this
# radius is not used in the
# calculations inside the function.
print(Cylinder(5, 10))
```

Getting Help in R

- built in help system
  - help.start()
  - help
    - Eg.
      - help(plot)
      - ?plot
- Online
  - R Bloggers
  - Stack Overflow
  - Twitter
  - RStudio Community

Quitting R Studio

- Restart session()
  - .rs.restartR()
- Example 1: Terminate an R Session Using quit() Function
  - quit()
- Example 2: Don’t Save Workspace When Using quit() Function
  - quit(save = "no")

Installing and loading packages

- Install R packages from
  - CRAN
  - GitHub
  - BitBucket
  - Bioconductor
  - rForge
- Packages from CRAN can be installed using install.packages()
- GitHub
  - devtools::install_github("tidyverse/ggplot2")
  - remotes::install_github("tidyverse/dplyr")

Problem

- How to install packages from BitBucket,Bioconductor,rForge

Data structures

- A data structure is a particular way of organizing data

The most essential data structures used in R include:

- Vectors
- Lists
- Dataframes
- Matrices
- Arrays
- Factors
- Tibbles

```r
# R program to illustrate Vector

# Vectors(ordered collection of same data type)
X = c(1, 3, 5, 7, 8)

# Printing those elements in console
print(X)
```

Lists

- A list is a generic object consisting of an ordered collection of objects.

```
# R program to illustrate a List

# The first attributes is a numeric vector  containing the employee IDs which is  created using the 'c' command here
empId = c(1, 2, 3, 4)

# The second attribute is the employee name which is created using this line of code here which is the character vector
empName = c("Debi", "Sandeep", "Subham", "Shiba")

# The third attribute is the number of employees which is a single numeric variable.
numberOfEmp = 4

# We can combine all these three different data types into a list containing the details of employees
# which can be done using a list command
empList = list(empId, empName, numberOfEmp)

print(empList)
```

Dataframes

- Dataframes are generic data objects of R which are used to store the **tabular data.**

```
# R program to illustrate dataframe

# A vector which is a character vector
Name = c("Amiya", "Raj", "Asish")

# A vector which is a character vector
Language = c("R", "Python", "Java")

# A vector which is a numeric vector
Age = c(22, 25, 45)

# To create dataframe use data.frame command
# and then pass each of the vectors
# we have created as arguments
# to the function data.frame()
df = data.frame(Name, Language, Age)

print(df)
```

Matrices

```
# R program to illustrate a matrix
 # Taking sequence of elements
A = matrix(
c(1, 2, 3, 4, 5, 6, 7, 8, 9),
  nrow = 3, ncol = 3,                 # No of rows and columns
  byrow = TRUE   )
                                # By default matrices are
                                # in column-wise order
                               # So this parameter decides
                               # how to arrange the matrix
print(A)
```

Arrays

Arrays are the R data objects which store the data in more than two dimensions

```
# R program to illustrate an array

A = array(c(1, 2, 3, 4, 5, 6, 7, 8),
          dim = c(2, 2, 2)    )
                    # Taking sequence of elements

                    # Creating two rectangular matrices
                    # each with two rows and two columns
print(A)
```

Factors

Factors are the data objects which are used to categorize the data and store it as levels.

```
# R program to illustrate factors


# Creating factor using factor()
fac = factor(c("Male", "Female", "Male",
               "Male", "Female", "Male", "Female"))


print(fac)
```

R Variables – Creating, Naming and Using Variables in R

- A variable is a memory allocated for the storage of specific data and the name associated with the variable is used to work around this reserved block.
- R Programming Language is a dynamically typed language
- **Creating Variables in R Language**
  - Using equal to operators
    variable_name = value
  - using leftward operator
    variable_name <- value
  - using rightward operator
    value -> variable_name

eg

```
# R program to illustrate
# Initialization of variables

# using equal to operator
var1 = "hello"
print(var1)

# using leftward operator
var2 <- "hello"
print(var2)

# using rightward operator
"hello" -> var3
print(var3)
```

Contd…

- Nomenclature of R Variables
- Methods for R Variables
  - class(variable)
  - Eg:

    ```
    var1 = "hello"
    print(class(var1))
    ```
  - ls()
  - rm(a)
- Scope of Variables in R programming
  - Global
  - Local
  - Dynamic scoping

```
z<-3
f <- function(x, y)
{
x^2 + y/z
# z has dynamic scoping
}
```

R Data Types

- **R Data types** are used to specify the kind of data that can be stored in a variable.

Basic Data Types	Values	Examples
Numeric	Set of all real numbers	"numeric_value <- 3.14"
Integer	Set of all integers, Z	"integer_value <- 42L"
Logical	TRUE and FALSE	"logical_value <- TRUE"
Complex	Set of complex numbers	"complex_value <- 1 + 2i"
Character	“a”, “b”, “c”, …, “@”, “#”, “$”, …., “1”, “2”, …etc	character_value <- "Hello Geeks"

Example

```
# A simple R program
# to illustrate Numeric data type

# Assign an integer value to y
y = 5

# print the class name of variable
print(class(y))

# print the type of variable
print(typeof(y))
```

Output

```
[1] “numeric"
[1] "double"
```

Numeric Data type in R

```
# A simple R program
# to illustrate Numeric data type


# Assign a decimal value to x
x = 5.6


# print the class name of variable
print(class(x))


# print the type of variable
print(typeof(x))
```

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

Contd…

```
# A simple R program
# to illustrate Numeric data type

# Assign a integer value to y
y = 5

# is y an integer?
print(is.integer(y))
```

```
[1] FALSE
```

Integer Data type in R

```
# A simple R program
# to illustrate integer data type


# Create an integer value
x = as.integer(5)


# print the class name of x
print(class(x))


# print the type of x
print(typeof(x))


# Declare an integer by appending an L suffix.
y = 5L


# print the class name of y
print(class(y))


# print the type of y
print(typeof(y))
```

```
[1] "integer"
[1] "integer"
[1] "integer"
[1] "integer"
```

Logical Data type in R

```
# A simple R program
# to illustrate logical data type


# Sample values
x = 4
y = 3


# Comparing two values
z = x > y


# print the logical value
print(z)


# print the class name of z
print(class(z))


# print the type of z
print(typeof(z))
```

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

Character Data type in R

```
# A simple R program
# to illustrate character data type


# Assign a character value to char
char = "Geeksforgeeks"


# print the class name of char
print(class(char))


# print the type of char
print(typeof(char))
```

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

Find Data Type of an Object in R

```r
# A simple R program
# to find data type of an object

# Logical
print(class(TRUE))

# Integer
print(class(3L))

# Numeric
print(class(10.5))

# Complex
print(class(1+2i))

# Character
print(class("12-04-2020"))
```

```
[1] "logical"
[1] "integer"
[1] "numeric"
[1] "complex"
[1] "character"
```

Type verification

```
# A simple R program
# Verify if an object is of a certain datatype

# Logical
print(is.logical(TRUE))

# Integer
print(is.integer(3L))

# Numeric
print(is.numeric(10.5))

# Complex
print(is.complex(1+2i))

# Character
print(is.character("12-04-2020"))

print(is.integer("a"))

print(is.numeric(2+3i))
```

```
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] TRUE
[1] FALSE
[1] FALSE
```

Coerce or Convert the Data Type of an Object to Another

```
# A simple R program
# convert data type of an object to another


# Logical
print(as.numeric(TRUE))


# Integer
print(as.complex(3L))


# Numeric
print(as.logical(10.5))


# Complex
print(as.character(1+2i))


# Can't possible
print(as.numeric("12-04-2020"))
```

```
[1] 1
[1] 3+0i
[1] TRUE
[1] "1+2i"
[1] NA
Warning message:
In print(as.numeric("12-04-2020")) : NAs
introduced by coercion
```

Sorting of Arrays in R Programming

```
# create a linear array
arr <- c(9, 8, 7, 6, 5, 4, 3, 2, 1)
```

```
[1] 1 2 3 4 5 6 7 8 9
```

```
# use of sort function to sort array
# by default it is sorted in increasing order
sort(arr)
```

```
# create linear array
arr <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
```

```
[1] 9 8 7 6 5 4 3 2 1
```

```
# use in built sort function to sort in decreasing order
sort(arr, decreasing = TRUE)
```

The major drawback of the sort() function is that it cannot sort data frames.

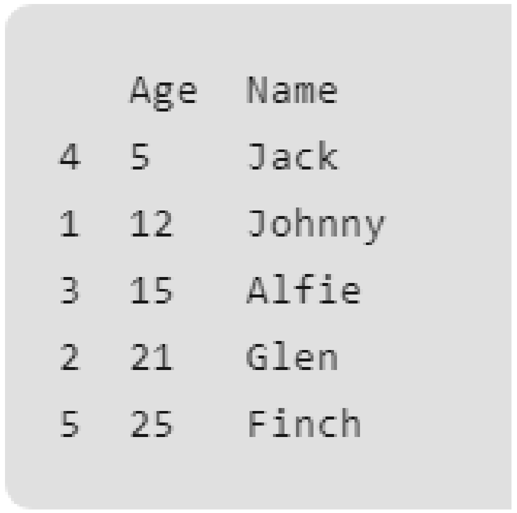
order() function

```
# define dataframe
df <- data.frame("Age" = c(12, 21, 15, 5, 25),
                 "Name" = c("Johnny", "Glen", "Alfie",
                            "Jack", "Finch"))


# sort the dataframe on the basis of
# age column and store it in newdf
newdf <- df[order(df$Age), ]


# print sorted dataframe
print(newdf)
```

Output:

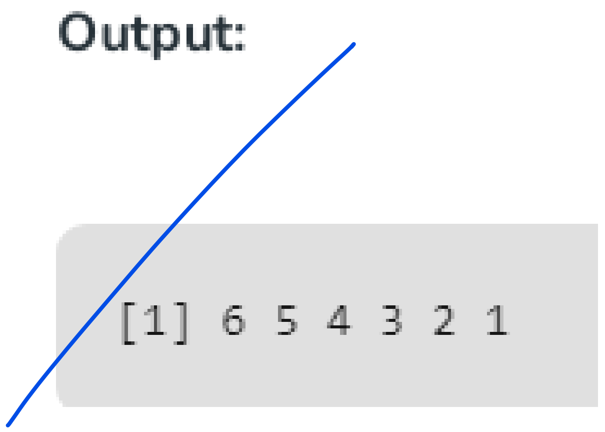


```
   Age  Name
4  5    Jack
1  12   Johnny
3  15   Alfie
2  21   Glen
5  25   Finch
```

Order()

```
# define vector
r = c(10, 20, 30, 40, 50, 60)

# sort in decreasing order
order(-r)
```



Output:

```
[1] 6 5 4 3 2 1
```

Problem

- Write a R program to Sort array using the loop
- Write a R program to Sort string array without using the loop

The use of dplyr package

- arrange()

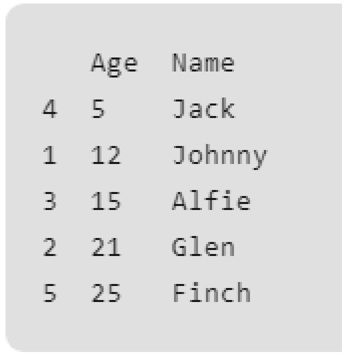
```
# install package dplyr
install.packages("dplyr")

# import library dplyr
library(dplyr)

# create dataframe
df <- data.frame("Age" = c(12, 21, 15, 5, 25),
                                                            "Name" = c("Johnny", "Glen", "Alfie",
                                                                                                  "Jack", "Finch"))


# sort the dataframe on the basis of
# age column using arrange method
arrange(df,age)
```

Output:



```
   Age  Name
4  5    Jack
1  12   Johnny
3  15   Alfie
2  21   Glen
5  25   Finch
```

Ordering Factor Values in R

- library(dplyr)
- as.ordered(factor_data)

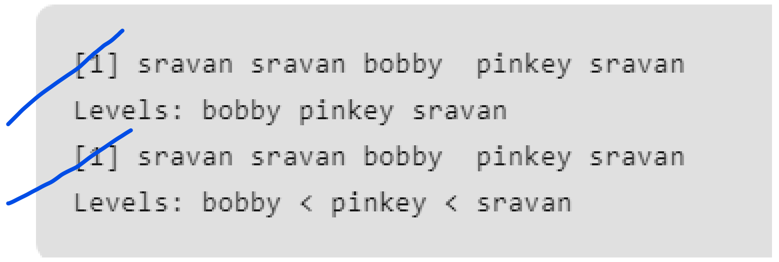
```
library(dplyr)

# create factor data with 5 strings
factor_data < - as.factor(c("sravan", "sravan", "bobby",
"pinkey", "sravan"))

# display before ordering
print(factor_data)

# display after ordering
print(as.ordered(factor_data))
```

Output:

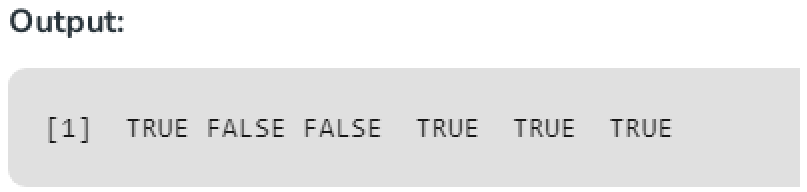


```
[1] sravan sravan bobby  pinkey sravan
Levels: bobby pinkey sravan
[1] sravan sravan bobby  pinkey sravan
Levels: bobby < pinkey < sravan
```

Handling Missing Values in R Programming

- Dealing Missing Values in R
  - is.na() Function for Finding Missing values

```
x<- c(NA, 3, 4, NA, NA, NA)
is.na(x)
```


Output:

```
[1]  TRUE FALSE FALSE  TRUE  TRUE  TRUE
```

- **na.omit**– omits every row containing even one NA
- **na.fail**– halts and does not proceed if NA is encountered
- **na.exclude**– excludes every row containing even one NA but keeps a record of their original position
- **na.pass**– it just ignores NA and passes through it

Problem

- Create a data frame with student name, usn,cgpa.
- Leave some values as NA in your data frame
- Display the rows not having NA
- Halt your display of code if NA is encountered
- Print by Excluding every row containing even one NA but keeps a record of their original position
- Ignores NA and print the data frame

Problems

- Write a R script using function to store student details(usn,name, 6 subjects marks) using

a) Vectors

   i) Find out their total marks and average. ii) Check whether they are pass or fail in the all subjects using logical & relational operator.

b) Lists

c) Dataframes

d) Matrices

e) Arrays

f) Factors

Problem

- Write a R script to store faculty details(name, fid,salary, no. of papers published, no of books written, no of patents published, no. of consultancy works ,no of funded projects)using

a) Vectors
   - Give weightage for their contributions(eg. For each papers published 5 points)
   - Find out the faculty total points for their contributions.
   - If they score >75 display that “Appraisal is good” else “not satisfactory”

b) Lists

c) Dataframes

d) Matrices

e) Arrays

f) Factors

References

- https://intro-r.rsquaredacademy.com/getting-help-in-r
- http://countbio.com/web_pages/left_object/R_for_biology/R_fundamentals/R_sessions.html
- https://intro-r.rsquaredacademy.com/install-update-r-packages
- https://www.geeksforgeeks.org/r-data-types/?ref=lbp