BT307 Biological Data Analysis Lab Assignment No. 1 25.01.2024

Note: Submit the assignment online through Moodle either in .doc or .pdf format. Your final report file should be named as "**YourName_BT307_Lab1_25012024**". Make sure that your name and roll numbers are written at the first page of your final report. Note that you can upload only one file; thus, put together all the answers in a single file.

Goal of this exercise is to learn about the basic functions of data types in R.

(1) Assign different types of data

```
x <- 13.4                 # numeric
y <- 10L                  # integer
z <- 5+3i                 # complex
p <- "R is an interesting programming language"     # character
q <- TRUE                 #logical
```

(2) Print the values and their types

```
x; class(x); typeof(x); attributes(x)
y; class(y); typeof(y); attributes(y)
z; class(z); typeof(z); attributes(z)
p; class(p); typeof(p); attributes(p)
q; class(q); typeof(q); attributes(q)
```

(3) Assign a series of values and print them

```
x <- 1:10
x; typeof(x); length(x)
```

(4) One can convert from one type to another as convert from numeric to integer

```
#convert from numeric to integer
a <- as.numeric(x)
a; class(a); typeof(a)

#convert from integer to numeric
b <- as.numeric(y)
b; class(b)
```

(5) Create a raw vector

```
r <- as.raw(c(0x1, 0x2, 0x3, 0x4, 0x5))
r
```

(6) Verify if an object is of a certain datatype

```
print(is.logical(TRUE))              # Logical
print(is.integer(3L))                # Integer
print(is.numeric(10.5))              # Numeric
print(is.complex(1+2i))              # Complex
print(is.character("12-04-2020"))    # Character
print(is.integer("a"))
print(is.numeric(2+3i))

#Logical Data type
variable_logical<- TRUE
cat(variable_logical,"\n")
```

```
cat("The data type of variable_logical is",class(variable_logical),"\n")
```

(7) Vectors: A vector is the most common and basic data structure in R

```
vector()
vector("character", length = 5)
character(5)
numeric(5)
logical(5)
```

(8) One can also create vectors by directly specifying their content.

```
x <- c(1,2,3)
x; class(x); typeof(x)
x1 <- c(1L,2L,3L)
x1; class(x1); typeof(x1)
y <- as.integer(x)
y; class(y); typeof(y)
```

(9) Examining vectors

```
z <- c("Sumit Ahire", "Sumit Kumar", "Sumit Nayan")
z; class(z); typeof(z); length(z); str(z)
```

(10) Adding elements to vectors

```
z <- c(z, "Sumit Shankar")
z; class(z); typeof(z); length(z); str(z)
```

(11) Vectors from a sequence of numbers

```
x <- 1:10
y <- seq(10)
x; y
z <- seq(from=1, to=100, by=5)
z; class(z); typeof(z); length(z); str(z)
```

(12) Missing Data: R supports missing data in vectors, they are represented as NA.

```
x <- c(0.5, NA, 0.7)
x; class(x); typeof(x)
x <- c(TRUE, FALSE, NA)
x; class(x); typeof(x)
x <- c("a", NA, "c", "d", "e")
x; class(x); typeof(x)
x <- c(1+5i, 2-3i, NA)
x; class(x); typeof(x)
```

(13) The function *is.na()* indicates the elements of the vectors that represent missing data, and the function *anyNA()* returns TRUE if the vector contains any missing values:

```
x <- c("a", NA, "c", "d", NA)
y <- c("a", "b", "c", "d", "e")
is.na(x); is.na(y)
anyNA(x); anyNA(y)
```

(14) What Happens When You Mix Types Inside a Vector?

```
x1 <- c(1.7, "a")
x1; class(x1)
x2 <- c(TRUE, 2)
x2; class(x2)
x3 <- c("a", TRUE)
x3; class(x3)
```

(15) One can control how vectors are coerced explicitly using the as.<class_name>() functions:

```
y1 <-as.numeric(x1)
y1
is.na(y1)
```

(16) Objects Attributes: objects can have attributes, for example (class, attributes, names, dimnames, dim, length, nchar)

```
names(x1) <- c('Numeric', 'Character')
x1
head(airquality)
names(airquality)

# dimnames()
x4 <- array(1:12)
x4
x4 <- array(1:12, dim = c(1,2,2), dimnames = list(c("C1"), c("R1", "R2"), c("M1", "M2")))
x4
```