LABWORK – 2

Introduction to R Data Types

LISTS:

1. **Creating a list using the list ( ) function**

```
> myList <- list (matrix (c (1, 3, 5, 7), nrow = 2, byrow = TRUE), "beu", c (2, 4, 6, 8, 10), 3.14)        # Creating a list which contains matrix, numbers > strings and vectors as its elements
myList
[[1]]
     [,1] [,2]
[1,]    1    3
[2,]    5    7
[[2]]
[1] "beu"
[[3]]
[1]  2  4  6  8 10
[[4]]
[1] 3.14
```

2. **Creating an empty list and then populating it**

```
> myList1 <- list()
> myList1 [[1]] <- matrix (c (2,4,6,8), nrow = 2)
> myList1 [[2]] <- "beu"
> myList1 [[3]] <- 3.14
> myList1 [[4]] <- c (1,3,5,7,9)
> myList1
[[1]]
     [,1] [,2]
[1,]    2    6
[2,]    4    8
[[2]]
[1] "beu"
[[3]]
[1] 3.14
[[4]]
[1] 1 3 5 7 9
```

3. **Creating list elements with names**

```
> myList <- list (myMatrix = matrix (c (1, 3, 5, 7), nrow = 2, byrow = TRUE), myVector = c (2, 4, 6, 8, 10), myString = "beu", myNumber = 3.14 )
> myList
$myMatrix
     [,1] [,2]
[1,]    1    3
[2,]    5    7
$myVector
[1]  2  4  6  8 10
$myString
[1] "beu"
$myNumber
[1] 3.14
```

4. **Naming/Alter names of the elements of the list using names function.**

```
> names (myList) <- c ("myNewMatrix", "myNewVector", "myNewString", "myNewNumber")
> myList
$myNewMatrix
     [,1] [,2]
[1,]    1    3
[2,]    5    7
$myNewVector
[1]  2  4  6  8 10
$myNewString
[1] "beu"
$myNewNumber
[1] 3.14
```

5. **Structure of a list**

```
> str(myList)                                   # Using str ( ) function
List of 4
 $ myMatrix: num [1:2, 1:2] 1 5 3 7
 $ myVector: num [1:5] 2 4 6 8 10
 $ myString: chr "beu"
 $ myNumber: num 3.14
```

6. Accessing List Elements

```
> myList
$myMatrix
     [,1] [,2]
[1,]    1    3
[2,]    5    7

$myString
[1] "beu"

$myVector
[1]  2  4  6  8 10

$myNumber
[1] 3.14
```

Difference between myList[[*n*]] and myList[*n*]

```
> myList[[1]]        # This is an element, not a list. It is the 1st element of myList
     [,1] [,2]
[1,]    1    3
[2,]    5    7
> myList[1]          #This is a list, not an element. The list contains one element,
$myMatrix            taken from the 1st element of myList.
     [,1] [,2]
[1,]    1    3
[2,]    5    7
> myList [c (2,4)]          # sublist of myList which contains 2nd and 3rd elements
$myVector
[1]  2  4  6  8 10
$myNumber
[1] 3.14
```

7. Accessing list elements by Name

```
> myList [["myNumber"]]        # Accessing the element that is named
[1] 3.14                       'myNumber'
> myList ["myNumber"]          # Accessing a sublist that contains only one
$myNumber                      element whose name is 'myNumber'
[1] 3.14
> myList [c("myNumber","myString")]        # Selecting sublist extracted from
$myNumber                                  myList
[1] 3.14
$myString
[1] "beu"
```

8. **Removing element from list**

```
> myList [["myNumber"]] <- NULL          # Assingning NULL to the element
> myList                                   myNumber
$myMatrix
     [,1] [,2]
[1,]    1    3
[2,]    5    7
$myVector
[1]  2  4  6  8 10
$myString
[1] "beu"

> myList [c("myMatrix","myString")] <- NULL
> myList
$myVector
[1]  2  4  6  8 10
```

9. **Removing element from list**

```
> myList
$myMatrix
     [,1] [,2]
[1,]    1    3
[2,]    5    7

$myVector
[1]  2  4  6  8 10

> myList2
[[1]]
[1] 22

[[2]]
[1] "CUTN"

[[3]]
NULL

> merged.list <- c(myList,myList2)
> print(merged.list)
$myMatrix
     [,1] [,2]
[1,]    1    3
[2,]    5    7

$myVector
[1]  2  4  6  8 10

[[3]]
[1] 22

[[4]]
[1] "CUTN"

[[5]]
NULL
```

10. Converting a list to a vector

```
> myList
$myMatrix
     [,1] [,2]
[1,]    1    3
[2,]    5    7

$myVector
[1]  2  4  6  8 10
```

```
> myVector <- unlist (myList)
```

myMatrix1	myMatrix2	myMatrix3	myMatrix4
1	5	3	7

myVector1	myVector2	myVector3	myVector4
2	4	6	8

myVector5
10

FACTORS:

1. **Creating a factor using the factor ( ) function**

```
> myData = c (1,2,2,4,6,3,1,2,3,4,6,6,1,3,1,2,3,3,1)# Creating a vector called myData
> myFactor = factor (myData)              # Creating a factor using factor function by
> myFactor                                  passing a vector into it
 [1] 1 2 2 4 6 3 1 2 3 4 6 6 1 3 1 2 3 3 1
Levels: 1 2 3 4 6
```

2. **Levels of the factor**

```
> levels(myFactor)          #The choices "1" "2" "3" "4" and "6" are the levels  of the
[1] "1" "2" "3" "4" "6"     factor and can be retrieved with the levels function
> nlevels(myFactor)         # provides us the length of the levels of the factor
[1] 5
```

3. **Changing factor levels**

```
# By specifying the levels arguments
> myData = c (1, 2, 2, 4, 6, 3, 1, 2, 3, 4, 6, 6, 1, 3, 1, 2, 3, 3, 1)
> myNewFactor = factor (myData, labels= c("One", "Two", "Three", "Four", "Six" ))
> myNewFactor
 [1] One   Two   Two   Four  Six   Three One   Two   Three Four  Six   Six
[13] One   Three One   Two   Three Three One
Levels: One Two Three Four Six
```

4. **Sorting**

```
> sort (myFactor)
 [1] 1 1 1 1 1 2 2 2 2 3 3 3 3 3 4 4 6 6 6
Levels: 1 2 3 4 6
> myFactor1 <- factor (myData, levels = unique (myData))
> myFactor1
 [1] 1 2 2 4 6 3 1 2 3 4 6 6 1 3 1 2 3 3 1
Levels: 1 2 4 6 3      #here the order of the levels matches the order of the first
                          appearance in the data
```

5. **Generating factor levels using gl (a, b) function**

```
# where ‘a’ is the integers for numbers of levels and ‘b’ is an integer for the number of
times each level has repeated.
> gl (3, 2)
[1] 1 1 2 2 3 3
Levels: 1 2 3
> gl (4, 3)
 [1] 1 1 1 2 2 2 3 3 3 4 4 4
Levels: 1 2 3 4
```

```
> gl(3, 2, labels = c("mango", "berry", "apple"))
[1] mango mango berry berry apple apple
Levels: mango berry apple
```

6. **Combining factors**

```
> myAddons <- gl(4, 2, labels = c("Nuts", "Cheese", "Olives", "Pepperoni"))
> myBase <- gl (2,1,8, labels = c("Thin_crust", "Stuffed_crust"))
> myPizza <- interaction (myAddons,myBase)
> myPizza
[1] Nuts.Thin_crust         Nuts.Stuffed_crust      Cheese.Thin_crust
[4] Cheese.Stuffed_crust    Olives.Thin_crust       Olives.Stuffed_crust
[7] Pepperoni.Thin_crust    Pepperoni.Stuffed_crust
8 Levels: Nuts.Thin_crust Cheese.Thin_crust ... Pepperoni.Stuffed_crust
```

DATA FRAMES:

1. **Creating a dataframe using the data.frame ( ) function**

```
> myDataFrame <- data.frame (items = c ("apple", "carrot", "tomato"), cost_per_item
= c(110,60,25))
> myDataFrame
      items      cost_per_item
1     apple      110
2     carrot     60
3     tomato     25
```

2. **Class of the object created**

```
> class(myDataFrame)
[1] "data.frame"
```

3. **Inspecting data frames**

```
> rownames(myDataFrame)
[1] "1" "2" "3"

> colnames(myDataFrame)
[1] "items"        "cost_per_item"

> dimnames(myDataFrame)
[[1]]
[1] "1" "2" "3"
[[2]]
[1] "items"        "cost_per_item"

> nrow(myDataFrame)
[1] 3

> ncol(myDataFrame)
[1] 2

> dim(myDataFrame)
[1] 3 2

> length (myDataFrame)
[1] 2

> names (myDataFrame)
[1] "items"        "cost_per_item"
```

4. Indexing data frames

```
> myDataFrame
      items       cost_per_item
1     apple       110
2     carrot      60
3     tomato      25

> myDataFrame[,2]                 # Extracting 2nd column
[1] 110  60  25

> myDataFrame[2,]                 # Extracting 2nd row
  items cost_per_item
2 carrot          60

> myDataFrame["items"]
      items
1     apple
2     carrot
3     tomato

> myDataFrame[1:2]

      items       cost_per_item
1     apple       110
2     carrot      60
3     tomato      25

> myDataFrame[1:2,1]
[1] apple  carrot
Levels: apple carrot tomato

> myDataFrame[1:2,1, drop=FALSE]
      items
1     apple
2     carrot
```