apply(), lapply(), sapply(), tapply() and mapply() functions

Apply() function:

•The apply() function can be feed with many functions to perform redundant application on a collection of object (data frame, list, vector, etc.).
•The **purpose** of apply() is primarily **to avoid explicit uses of loop constructs**. They can be used for an input list, matrix or array and apply a function.
•Any function can be passed into apply().

Different functions are:

apply() function
lapply( )function
sapply() function
tapply() function

apply() function

apply() takes Data frame or matrix as an input and gives output in vector, list or array.
apply() Function is primarily used to avoid explicit uses of loop constructs.

Syntax:
This function takes 3 arguments:

apply(X, MARGIN, FUN)

Here: -x: an array or matrix
MARGIN: take a value or range between 1 and 2 to define where to apply the function
MARGIN=1: the manipulation is performed on rows
MARGIN=2: the manipulation is performed on columns
MARGIN=c(1,2) the manipulation is performed on rows and columns
FUN: tells which function to apply. Built functions like **mean, median, sum, min, max** and **even user-defined functions can be applied**

```
>  m1 <- matrix(C<-(1:15), nrow=5, ncol=6)
> m1
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    6   11    1    6   11
[2,]    2    7   12    2    7   12
[3,]    3    8   13    3    8   13
[4,]    4    9   14    4    9   14
[5,]    5   10   15    5   10   15

> a_m1 <- apply(m1, 2, sum)
> a_m1
[1] 15 40 65 15 40 65

> a_m1 <- apply(m1, 2, mean)
> a_m1
[1]  3  8 13  3  8 13
```

lapply() function

•**lapply()** function is useful for performing operations on list objects and **returns a list object of same length of original set.**

•lappy() returns a list of the similar length as input list object, **each element of which is the result of applying FUN to the corresponding element of list**.

•lapply() takes list, vector or data frame as input and gives output in list.

lapply(X, FUN)

Arguments:

-X: A vector or an object

-FUN: Function applied to each element of x

Difference between apply() and lapply()

l in lapply() stands for list.

The difference between lapply() and apply() lies between the output return.

The output of lapply() is a list.

lapply() can be used for other objects like data frames and lists.

lapply() function does not need MARGIN.

```
>cnames <- c("INDIA","AUSTRALIA","CHIANA","NEPAL")   # Create vector named cnames

> Cnames                                              # print
[1] "INDIA"    "AUSTRALIA" "CHIANA"    "NEPAL“

> str(cnames)                                         # print structure of cnames
 chr [1:4] "INDIA" "AUSTRALIA" "CHIANA" "NEPAL“

> cnames_lower <-lapply(cnames, tolower)     # apply function tolower on cnames
> str(cnames_lower)                          # print structure of cnames_lower
List of 4
 $ : chr "india"
 $ : chr "australia"
 $ : chr "chiana"
 $ : chr "nepal“


> cnames_lower <-unlist(lapply(cnames,tolower))          #unlist cnames
> str(cnames_lower)                          # print structure of cnames_lower
 chr [1:4]  "india" "australia" "chiana" "nepal”
```

sapply() function

sapply() function takes list, vector or data frame as input and gives output in vector or matrix.
It is **useful for operations on list objects** and **returns a list object of same length** of original set.
sapply() function does the same job as lapply() function but returns a vector.

Syntax:

sapply(X, FUN)
Arguments:
-X: A vector or an object
-FUN: Function applied to each element of x

# cars is an inbuilt databse. Structure of cars is:

```
> str(cars)
'data.frame':   50 obs. of  2 variables:
 $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
 $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

Example:

```
> dt <- cars
> lmn_cars <- lapply(dt, min)
>smn_cars <- sapply(dt, min)

> lmn_cars
$speed
[1] 4

$dist
[1] 2 # Output is in the form of list

> smn_cars
speed  dist
    4     2
 # Output is in the form of vector
```

Example:

```
> dt <- cars
> lmn_cars <- lapply(dt, max)
>smn_cars <- sapply(dt, max)

> lmn_cars
$speed
[1] 25

$dist
[1] 120 # Output is in the form of list

> smn_cars
speed  dist
   25    120
 # Output is in the form of vector
```

sapply() function is more efficient than lapply() in the output returned because sapply() store values directly into a vector.

Use of built-in function:
We can use a user built-in function into lapply() or sapply().

Example: We create a function named avg to compute the average of the minimum and maximum of the vector.

```
>avg <- function(x)  { ( min(x) + max(x) ) / 2}

>fcars <- sapply(dt, avg)

> fcars
speed  dist
 14.5  61.0
```

Difference between apply(), sapply() and lapply() :

Function	Arguments	Objective	Input	Output
apply	apply(x, MARGIN, FUN)	Apply a function to the rows or columns or both	Data frame or matrix	vector, list, array
lapply	lapply(X, FUN)	Apply a function to all the elements of the input	List, vector or data frame	list
sapply	sapply(X FUN)	Apply a function to all the elements of the input	List, vector or data frame	vector or matrix

tapply() function

tapply() computes a measure (mean, median, min, max, etc..) or a function for each factor variable in a vector.

It is a very useful function that lets you create a subset of a vector and then apply some functions to each of the subset.

Syntax:

tapply(X, INDEX, FUN = NULL)

Arguments:

-X: An object, usually a vector

-INDEX: A list containing factor

-FUN: Function applied to each element of x

```
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor", “virginica” ..: 1 1 1 1 1 1 1 1 1 1 ..

> data(iris)
> tapply(iris$Sepal.Width, iris$Species, median)
    setosa versicolor  virginica
       3.4        2.8        3.0
```

The mapply() Function:

The mapply() function stands for ‘multivariate’ apply. Its **purpose** is to **be able to vectorize arguments to a function that is not usually accepting vectors as arguments.**

In short, mapply() applies a Function to Multiple List or multiple Vector Arguments.

```
> Q1 <- matrix(c(rep(1, 4), rep(2, 4), rep(3, 4), rep(4, 4)),4,4)
> print(Q1)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    1    2    3    4
[3,]    1    2    3    4
[4,]    1    2    3    4
```

```
> Q2 <- mapply(rep,1:4,4)
> print(Q2)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    1    2    3    4
[3,]    1    2    3    4
[4,]    1    2    3    4
```

```
>Q2<-mapply(rep, 1:5, 5:1)
>Q2
```

Output:

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