apply()

Use the **apply()** function when you want to apply a function to the rows or columns of a matrix or data frame.

The basic syntax for the apply() function is as follows:

apply(X, MARGIN, FUN)

- X is the name of the matrix or data frame
- MARGIN indicates which dimension to perform an operation across (1 = row, 2 = column)
- FUN is the specific operation you want to perform (e.g. min, max, sum, mean, etc.)

The following code illustrates several examples of **apply()** in action.

```
#create a data frame with three columns and five rows
data <- data.frame(a = c(1, 3, 7, 12, 9),
                   b = c(4, 4, 6, 7, 8),
                   c = c(14, 15, 11, 10, 6))
data

#   a b  c
#1  1 4 14
#2  3 4 15
#3  7 6 11
#4 12 7 10
#5  9 8  6

#find the sum of each row
apply(data, 1, sum)

#[1] 19 22 24 29 23

#find the sum of each column
apply(data, 2, sum)

# a  b  c
#32 29 56

#find the mean of each row
apply(data, 1, mean)

#[1] 6.333333 7.333333 8.000000 9.666667 7.666667
```

```
#find the mean of each column, rounded to one decimal place
round(apply(data, 2, mean), 1)

#  a    b     c
#6.4  5.8  11.2

#find the standard deviation of each row
apply(data, 1, sd)

#[1] 6.806859 6.658328 2.645751 2.516611 1.527525

#find the standard deviation of each column
apply(data, 2, sd)

#       a        b        c
#4.449719 1.788854 3.563706
```

lapply()

Use the **lapply()** function when you want to apply a function to each element of a list, vector, or data frame and obtain a list as a result.

The basic syntax for the lapply() function is as follows:

lapply(X, FUN)

- X is the name of the list, vector, or data frame
- FUN is the specific operation you want to perform

The following code illustrates several examples of using **lapply()** on the columns of a data frame.

```
#create a data frame with three columns and five rows
data <- data.frame(a = c(1, 3, 7, 12, 9),
                   b = c(4, 4, 6, 7, 8),
                   c = c(14, 15, 11, 10, 6))
data

#   a b  c
#1  1 4 14
#2  3 4 15
#3  7 6 11
#4 12 7 10
#5  9 8  6
```

```
#find mean of each column and return results as a list
lapply(data, mean)

# $a
# [1] 6.4
#
# $b
# [1] 5.8
#
# $c
# [1] 11.2

#multiply values in each column by 2 and return results as a list
lapply(data, function(data) data*2)

# $a
# [1]  2  6 14 24 18
#
# $b
# [1]  8  8 12 14 16
#
# $c
# [1] 28 30 22 20 12
```

We can also use **lapply()** to perform operations on lists. The following examples show how to do so.

```
#create a list
x <- list(a=1, b=1:5, c=1:10)
x

# $a
# [1] 1
#
# $b
# [1] 1 2 3 4 5
#
# $c
# [1]  1  2  3  4  5  6  7  8  9 10

#find the sum of each element in the list
lapply(x, sum)

# $a
# [1] 1
#
# $b
# [1] 15
#
# $c
```

```
# [1] 55

#find the mean of each element in the list
lapply(x, mean)

# $a
# [1] 1
#
# $b
# [1] 3
#
# $c
# [1] 5.5

#multiply values of each element by 5 and return results as a list
lapply(x, function(x) x*5)

# $a
# [1] 5
#
# $b
# [1]  5 10 15 20 25
#
# $c
# [1]  5 10 15 20 25 30 35 40 45 50
```

sapply()

Use the **sapply()** function when you want to apply a function to each element of a list, vector, or data frame and obtain a **vector** instead of a list as a result.

The basic syntax for the sapply() function is as follows:

sapply(X, FUN)

- X is the name of the list, vector, or data frame
- FUN is the specific operation you want to perform

The following code illustrates several examples of using **sapply()** on the columns of a data frame.

```
#create a data frame with three columns and five rows
data <- data.frame(a = c(1, 3, 7, 12, 9),
                   b = c(4, 4, 6, 7, 8),
                   c = c(14, 15, 11, 10, 6))
data
```

```
#   a b  c
#1  1 4 14
#2  3 4 15
#3  7 6 11
#4 12 7 10
#5  9 8  6

#find mean of each column and return results as a vector
sapply(data, mean)

#  a   b    c
#6.4 5.8 11.2

#multiply values in each column by 2 and return results as a matrix
sapply(data, function(data) data*2)

#      a  b  c
#[1,]  2  8 28
#[2,]  6  8 30
#[3,] 14 12 22
#[4,] 24 14 20
#[5,] 18 16 12
```

We can also use **sapply()** to perform operations on lists. The following examples show how to do so.

```
#create a list
x <- list(a=1, b=1:5, c=1:10)
x

# $a
# [1] 1
#
# $b
# [1] 1 2 3 4 5
#
# $c
# [1]  1  2  3  4  5  6  7  8  9 10

#find the sum of each element in the list
sapply(x, sum)

# a  b  c
# 1 15 55

#find the mean of each element in the list
sapply(x, mean)

#  a   b   c
#1.0 3.0 5.5
```

tapply()

Use the **tapply()** function when you want to apply a function to subsets of a vector and the subsets are defined by some other vector, usually a factor.

The basic syntax for the tapply() function is as follows:

tapply(X, INDEX, FUN)

- X is the name of the object, typically a vector
- INDEX is a list of one or more factors
- FUN is the specific operation you want to perform

The following code illustrates an example of using **tapply()** on the built-in R dataset **iris.**

```
#view first six lines of iris dataset
head(iris)

#  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#1          5.1         3.5          1.4         0.2  setosa
#2          4.9         3.0          1.4         0.2  setosa
#3          4.7         3.2          1.3         0.2  setosa
#4          4.6         3.1          1.5         0.2  setosa
#5          5.0         3.6          1.4         0.2  setosa
#6          5.4         3.9          1.7         0.4  setosa

#find the max Sepal.Length of each of the three Species
tapply(iris$Sepal.Length, iris$Species, max)

#setosa versicolor  virginica
#   5.8        7.0        7.9

#find the mean Sepal.Width of each of the three Species
tapply(iris$Sepal.Width, iris$Species, mean)

# setosa versicolor virginica
#  3.428      2.770      2.974

#find the minimum Petal.Width of each of the three Species
tapply(iris$Petal.Width, iris$Species, min)

#  setosa versicolor virginica
#     0.1        1.0        1.4
```