R Online Editor - R Online IDE - R Coding Online (jdoodle.com)

Basic Data Types	Values
Numeric	Set of all real numbers
Integer	Set of all integers, Z
Logical	TRUE and FALSE
Complex	Set of complex numbers
Character	“a”, “b”, “c”, …, “@”, “#”, “$”, …., “1”, “2”, …etc

Numeric Datatype

Decimal values are called numerics in R. It is the default data type for numbers in R. If you assign a decimal value to a variable x as follows, x will be of numeric type.

```
x = 5.6

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

# print the type of variable
print(typeof(x))
[1] "numeric"
[1] "double"
```

When R stores a number in a variable, it converts the number into a “double” value or a decimal type with at least two decimal places. This means that a value such as “5” here, is stored as 5.00 with a type of double and a class of numeric. And also y is not an integer here can be confirmed with the **is.integer()** function.

```
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))
```

```
# assign integer value

var_x <- 27L
```

Logical Datatype

R has logical data types that take either a value of true or false. A logical value is often created via a comparison between variables.

```
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())
```

Complex Datatype

R supports complex data types that are set of all the complex numbers. The complex data type is to store numbers with an imaginary component.

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

 # Assign a complex value to x
 x = 4 + 3i

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

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

Character Datatype

R supports character data types where you have all the alphabets and special characters. It stores character values or strings. Strings in R can contain alphabets, numbers, and symbols. The easiest way to denote that a value is of character type in R is to wrap the value inside single or double inverted commas.

```
char = "R Programming"

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

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

A character object is used to represent string values in R. We convert objects into character values with the as.character() function:

> x = as.character(3.14)

> x # print the character string

[1] "3.14"

> class(x)

print the class name of x

[1] "character"

Two character values can be concatenated with the paste function.

> fname = "Joe"; lname ="Smith"

> paste(fname, lname)

[1] "Joe Smith"

To find the data type of an object you have to use **class()** function. The syntax for doing that is you need to pass the object as an argument to the function **class()** to find the data type of an object.

```
class(object)
```

Data Types

- Vectors
- Lists
- Matrices
- Arrays
- Factors
- Data Frames
-

- ## Vectors
  When you want to create vector with more than one element, you should use **c()**function which means to combine the elements into a vector.

```
# Create a vector.
apple <- c('red','green',"yellow")
print(apple)

# Get the class of the vector.
print(class(apple))
```

Lists

A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it.

```
Create a list.
list1 <- list(c(2,5,3),21.3,sin)

# Print the list.
print(list1)
```

Matrices

A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

```
# Create a matrix.
M = matrix( c('a','a','b','c','b','a'), nrow = 2, ncol = 3)
print(M)
```

Arrays

While matrices are confined to two dimensions, arrays can be of any number of dimensions. The array function takes a dim attribute which creates the required number of dimensions. In the below example we create an array with two elements which are 3x3 matrices each.

```
array(vector, dim = c(nrow, ncol, nmat))
```

Here,

- vector - the data items of same type
- nrow - number of rows
- ncol - number of columns
- nmat - the number of matrices of nrow * ncol dimension

```
# Create an array.
a <- array(c('green','yellow'),dim = c(3,3,2))
print(a)
```

```
, , 1

     [,1]    [,2]    [,3]
[1,] "green"  "yellow" "green"
[2,] "yellow" "green"  "yellow"
[3,] "green"  "yellow" "green"

, , 2

     [,1]    [,2]    [,3]
[1,] "yellow" "green"  "yellow"
[2,] "green"  "yellow" "green"
[3,] "yellow" "green"  "yellow"
```

```
array(c(1:15), dim = c(2,3,2))
```

- `c(1:12)` - a vector with values from **1** to**12**
- `dim = c(2,3,2)` - create two matrice of **2** by **3**dimension

```
, , 1

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

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
```

Factors

Factors are the r-objects which are created using a vector. It stores the vector along with the distinct values of the elements in the vector as labels

Data Frames

Data frames are tabular data objects. Unlike a matrix in data frame each column can contain different modes of data. The first column can be numeric while the second column can be character and third column can be logical. It is a list of vectors of equal length.

Data Frames are created using the **data.frame()** function.

```
  gender  height  weight  Age
1   Male   152.0      81   42
2   Male    171.5     93   38
3 Female   165.0     78   26
```

Variable Name

A valid variable name consists of letters, numbers and the dot or underline characters. The variable name starts with a letter or the dot not followed by a number.

Variable Assignment

The variables can be assigned values using leftward, rightward and equal to operator. The values of the variables can be printed using **print()** or **cat()** function. The **cat()**function combines multiple items into a continuous print output.

```
# Assignment using equal operator.
var.1 = c(0,1,2,3)

# Assignment using leftward operator.
var.2 <- c("learn","R")

# Assignment using rightward operator.
c(TRUE,1) -> var.3
```

```
print(var.1)
cat ("var.1 is ", var.1 ,"\n")
cat ("var.2 is ", var.2 ,"\n")
cat ("var.3 is ", var.3 ,"\n")
```

Mean, Median, Mode using R programming

```
x <- c(1,2,3,4,5,1,2,3,1,2,4,5,2,3,1,1,2,3,5,6)
mean(x)
median(x)
```

Mode: The most occurring number in the data set. For calculating mode, there is no default function in R. So, we have to create our own custom function.

Variance: How far a set of data values are spread out from their mean.

```
x <- c(1,2,3,4,5,1,2,3,1,2,4,5,2,3,1,1,2,3,5,6)
mean(x)
median(x)
var(x)
sqrt(var(x))
```

Output

[1] 2.8

[1] 2.5

[1] 2.484211

[1] 1.576138

To calculate quartile is a little thing in R. just used the quantile() function.

height=c(11,22,17,13,27,25,17,27,23,17,19,22,17,19,16,23,25,17,36,15,14,25,26,17,18,29,17,15,28,17)

print(quantile(height))

Output

```
  0%  25%  50%  75% 100%
11.0 17.0 18.5 25.0 36.0
```

Graphs in R

Graphs in R language is a preferred feature which is used to create various types of graphs and charts for visualizations. R language supports a rich set of packages and functionalities to create the graphs using the input data set for data analytics. The most commonly used graphs in the R language are scattered plots, box plots, line graphs, pie charts, histograms, and bar charts. R graphs support both two dimensional and three-dimensional plots for exploratory data analysis.There are R function like plot(), barplot(), pie() are used to develop graphs in R language. R package like ggplot2 supports advance graphs functionalities.

summary(mtcars)

hist(mtcars$mpg, col = "green")

```
hist(mtcars$mpg, col = "green") ## Plot 1
boxplot(mtcars$mpg, col="green")
barplot(table(mtcars$carb), col="green")
with(mtcars, plot(mpg, qsec))
```