R Data Types Part 1

Data types of Variables in R

variables in any programming language are used to store information in memory

Unlike other programming languages R variables are R-objects.

Types of R-Objects in R programming:

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

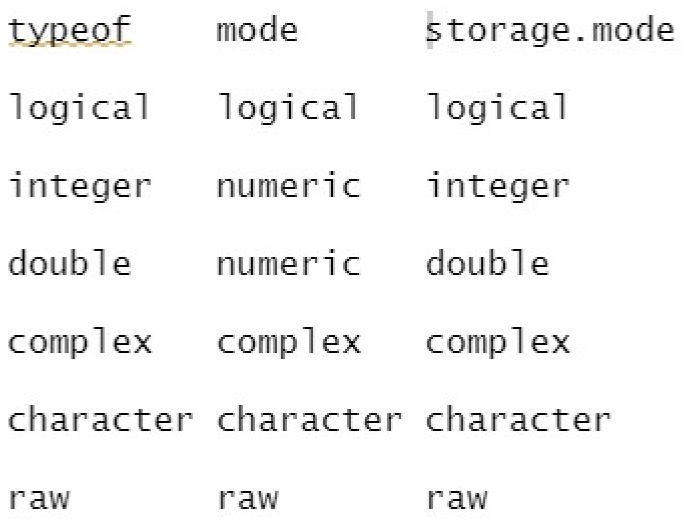
Simplest R-objects are vector objects.

There are 6 types of Atomic Vectors or Basic Vectors. (NOTE: There are basic types there are more classes in R)

atomic Vector types

Logical Integer Numeric Complex Character Raw

In R, type, mode and storage mode of a R-Object may be different. So, following is details for each types of atomic vectors.



typeof	mode	storage.mode
logical	logical	logical
integer	numeric	integer
double	numeric	double
complex	complex	complex
character	character	character
raw	raw	raw

Details for each types of atomic vectors

Create Integer type atomic vectors

Any whole number appended with L will be treated as integer by R Following is command to see all details of an integer variable Details include class, typesof, mode, and storage.mode

```
x=100000L
cat(x,class(x), typeof(x), mode(x), storage.mode(x))
```

```
## 100000 integer integer numeric integer
```

Create Numeric type atomic vectors

Any decimal point number or number (without L) will be treated as Numeric by R Following are commands to see all details of an Numeric variable Details include class, typesof, mode, and storage.mode

Numeric when number without suffix L

```
x=10
cat(x,class(x), typeof(x), mode(x), storage.mode(x))
```

```
## 10 numeric double numeric double
```

Numeric when number is decimal

```
x=10.67
cat(x,class(x), typeof(x), mode(x), storage.mode(x))
```

```
## 10.67 numeric double numeric double
```

Create Logical type atomic vectors

TRUE or FALSE will be treated as Logical by R Following are commands to see all details of an Logical variable Details include class, typesof, mode, and storage.mode

```
x=TRUE
cat(x,class(x), typeof(x), mode(x), storage.mode(x))
```

```
## TRUE logical logical logical logical
```

Create Complex type atomic vectors

Any number in form of a+bi will be treated as Complex by R Following are commands to see all details of an Complex variable Details include class, typesof, mode, and storage.mode

```
x=5+4i
cat(x,class(x), typeof(x), mode(x), storage.mode(x))
```

```
## 5+4i complex complex complex complex
```

Create Character type atomic vectors

Any single character or string will be treated as Character by R. R will give same type even if chracters are in single quotes or double quotes. Following are commands to see all details of an Character variable. Details include class, typesof, mode, and storage.mode

```
x="Hi"
cat(x,class(x), typeof(x), mode(x), storage.mode(x))
```

```
## Hi character character character character
```

Create Raw type atomic vectors

Any single character or string will be treated as Character by R. R will give same type even if chracters are in single quotes or double quotes. Following are commands to see all details of an Character variable. Details include class, typesof, mode, and storage.mode

```
x=charToRaw("Hello")
cat("bytes in Hello are :",x," details:",class(x), typeof(x), mode(x), storage.mode(x))
```

```
## bytes in Hello are : 48 65 6c 6c 6f  details: raw raw raw raw
```

Vectors

Using c() function to create vectors

In R we can use c() function also known as combine. It creates a vector

```
x_dim1 = c(10,20,30,40,50,60)
print(x_dim1)
```

```
## [1] 10 20 30 40 50 60
```

```
class(x_dim1)
```

```
## [1] "numeric"
```

Data type assignment after combine

Data types of returned value is based on highest hierarchy in all input values

Hierarchy is Logical < Integer < Numeric < Complex < Character

See output of following codes and note how the class / data types are assigned to vector

```
x_complex = c(3,0,TRUE,2+2i)
print(x_complex)
```

```
## [1] 3+0i 0+0i 1+0i 2+2i
```

```
class(x_complex)
```

```
## [1] "complex"
```

```
x_char = c(0,"IACSD R Prog",TRUE,2+2i)
print(x_char)
```

```
## [1] "0"            "IACSD R Prog" "TRUE"         "2+2i"
```

```
class(x_char)
```

```
## [1] "character"
```

Using vector function to create vectors

vector function creates an empty vector of given data type and size

Following code creates vector of numeric data type of size 5

```
#create vector of numeric data type of size 5
x_vec = vector("numeric", 5)
print(x_vec)
```

```
## [1] 0 0 0 0 0
```

```
#create vector of complex data type of size 5
x_vec = vector("complex", 5)
print(x_vec)
```

```
## [1] 0+0i 0+0i 0+0i 0+0i 0+0i
```

```
#create vector of logical data type of size 5
x_vec = vector("logical", 5)
print(x_vec)
```

```
## [1] FALSE FALSE FALSE FALSE FALSE
```

```
#create vector of character data type of size 5
x_vec = vector("character", 5)
print(x_vec)
```

```
## [1] "" "" "" "" ""
```

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.

```
x <- list(1, "a", c(1,2,3), 1+4i)
x
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] "a"
##
## [[3]]
## [1] 1 2 3
##
## [[4]]
## [1] 1+4i
```

list used just like key value pair

Every element of list can be assigned key and value

So, value can be accessed in format list$key

```
xlist <- list(a = "Shantanu Pathak", b = 1:10)
xlist
```

```
## $a
## [1] "Shantanu Pathak"
##
## $b
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Matrix

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

Matrix can be created by specifying number of rows and columns. “byrow = TRUE” means elements will be filled in row wise manner.

```
M = matrix( c('a','a','b','c','b','a'), nrow=2,ncol=3,byrow = TRUE)
print(M)
```

```
##      [,1] [,2] [,3]
## [1,] "a"  "a"  "b" 
## [2,] "c"  "b"  "a" 
```

ARRAY

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 dimension. In the below example we create an array with two elements which are 3x3 matrices each.

```
x_dim2 = array(c(10,20,30,40,50,60,70),dim=c(2,3))
print(x_dim2)
```

```
##      [,1] [,2] [,3]
## [1,]   10   30   50
## [2,]   20   40   60
```

Factors

factor- type vector contains a set of numeric codes with character-valued levels.

Regardless of the levels/labels of the factor, the numeric storage is an integer with 1 corresponding to the first level (in alph-order)

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. The labels are always character irrespective of whether it is numeric or character or Boolean etc. in the input vector. They are useful in statistical modeling.

```
students = factor(c(100,0,100,0,0,0), levels = c(0, 100), labels = c("boy", "girl"))
students
```

```
## [1] girl boy  girl boy  boy  boy 
## Levels: boy girl
```

Ordered factors

Ordinal variables are created using order=TRUE in factors

When we print the ordinal variable (ordered factor) in levels we can see the order mentioned

In following example Manager is assigned lowest level, the team lead ans then SME. This order is with respect to alphabetical order of values.

```
designation <- factor(c("SME", "Team Lead","Manager"), ordered =TRUE)
designation
```

```
## [1] SME       Team Lead Manager
## Levels: Manager < SME < Team Lead
```

DataFrame

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.

```
M1 = matrix(c(1,2,3,4),nrow=2)
#M1
df <- as.data.frame(M1)
df
```

V1 <dbl>	V2 <dbl>
1	3
2	4

2 rows