Data Types in R

Data Types in R

Numeric	Contains decimal as well as whole numbers
Integer	Contains only whole numbers
Character	Holds character strings
Factor	A vector that can contain only predefined values, and is used to store categorical data.
Logical	Can only take on two values, TRUE or FALSE.

Data Types in R

Vector	1 dimension	Sequence of data elements of the same basic type.
Matrix	2 dimensions	Like a Vector but additionally contains the dimension attribute.
Array	2 or more dimensions	Hold multidimensional data. Matrices are a special case of two-dimensional arrays.
Data frame	2 dimensions	Table-like data object allowing different data types for different columns.
List	Collection of data objects, each element of a list is a data object.	

Numeric

Numeric type object can store decimal as well as whole numbers.

Create two numeric objects x and y and assign values 4.5 and 3567 respectively.

```
x<-4.5
x
```

```
[1] 4.5
```

```
y<-3567
y
```

```
[1] 3567
```

```
class(x)
```

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

```
class(y)
```

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

class() function is used to check the data type of an object.

Numeric

R shows class of object x as numeric. To convert it to an integer format, use `as.integer()` function.

```
x<-as.integer(x)
x
```

```
[1] 4
```

So, new value of x is 4 after converting it into integer

```
class(x)
```

```
[1] "integer"
```

Integer

```
#To create an integer variable in R use as.integer() function.
```

```
f<-as.integer(22.5)
f
```

```
[1] 22
```

```
class(f)
```

```
[1] "integer"
```

```
x=8
class(x)
```

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

Note : The default class of an integer is a numeric class

Character

A character object is used to represent strings. A string is specified by using quotes (both single and double quotes will work). To convert any object into a character type, use **`as.character()`** function.

```
z<-"Welcome to R  Ready Reckon-er"
z
```

```
[1] "Welcome to R  Ready Reckon-er"
```

```
x<-"4.5"
x
```

```
[1] "4.5"
```

Any value enclosed in quotes is stored as a character object

```
class(z)
```

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

```
class(x)
```

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

Factor

Factor objects are used to categorize data and can store both strings and integers.

```
# Create an object x
x<-c("high", "medium", "low", "low", "medium", "high", "high", "high",
"medium", "low","low")
```

c() combines data of different types

```
# Check whether object x is a factor or character
is.factor(x)
```

```
[1] FALSE
```

is.factor() function returns True or False after checking whether the object is of type factor or not

```
is.character(x)
```

```
[1] TRUE
```

is.character() function returns True or False after checking whether the object is of type character or not

Factor

```
#Create a factor object using factor() function
```

```
x<-factor(x)
x
```

```
[1] high   medium low    low    medium high   high   high
[9] medium low    low
Levels: high low medium
```

- A factor is a categorical variable that can take only one of a fixed, finite set of possibilities. Those possible categories are the levels.
- Levels are the unique data values. Above output tells how many levels are there in x.
- Alternatively, one can use **level()** function to check levels.

```
levels(x)
```

```
[1] "high" "low" "medium"
```

Factor object x has 11 elements and 3 levels. By default the levels are sorted alphabetically

Factor

```
#To specify the order of the factor, use ordered() function
```

```
x_ordered<-ordered(x, levels=c("low", "medium","high"))
x_ordered
```

```
[1] high   medium low    low    medium high   high   high   medium low
low
```

```
Levels: low < medium < high
```

levels= takes the levels in the order in which you want them to be ordered

One of the most important uses of factors is in statistical modeling, since categorical variables enter into statistical models differently than continuous variables, storing data as factors insures that the modeling functions will treat such data correctly.

Logical

Logical type objects can only take 2 values i.e. TRUE or FALSE.

```
# Create an object x and assign a value 4.5 and check whether it is an integer
x<-4.5
is.integer(x)
[1] FALSE
```

is.integer() function checks whether the object is integer or not.

```
# Create two numeric objects y and z
# Check whether y is greater than z or not
y<-4
z<-7
Result <- y > z
Result
[1] FALSE
```

- With this kind of statement, you are asking R to evaluate the logical question "Is it true that y is greater than z?"
- The object(Result) storing the answe of above question is of type logical
- You can check the class of the object using class()

Vector

A Vector is a Sequence of data elements of the same basic type.
Create three vectors: numeric, character and logical. All the elements of a vector must be of the same type.

```
# Numeric vector
a <- c(1,2,5.3,6,-2,4)
a
```

```
[1]  1.0  2.0  5.3  6.0 -2.0  4.0
```

```
# Character vector
b <- c("one","two","three")
b
```

```
[1] "one"   "two"   "three"
```

```
# Logical vector
d<-c(4,24,6,4, 2,7)
d>5
```

```
[1] FALSE TRUE TRUE FALSE FALSE TRUE
```

Matrices

Matrix is a two-dimensional data structure in R. It similar to vectors but additionally contains the dimension attribute. To convert any object into a matrix type, use **`as.matrix()`** function.

Create a matrix with 3 rows and 2 columns.

```
x<-matrix(c(2, 3, 4, 5, 6, 7),nrow=3,ncol=2)
x
```

```
     [,1] [,2]
[1,]    2    5
[2,]    3    6
[3,]    4    7
```

- matrix() function is used to create a matrix.
- nrow= and ncol= is used to specify the dimension of the matrix

Note that the matrix is filled in by column-wise.

```
x<-matrix(c(2, 3, 4, 5, 6, 7),nrow=3,ncol=2,byrow=TRUE)
x
```

```
     [,1] [,2]
[1,]    2    3
[2,]    4    5
[3,]    6    7
```

byrow=TRUE fills the matrix row-wise

Matrices

It is possible to name the rows and columns of matrix during creation by passing a 2-element list to the argument **dimnames**.

```
x<-matrix(c(2, 3, 4, 5, 6, 7),nrow=3,ncol=2,byrow=TRUE,
          dimnames=list(c("X","Y","Z"), c("A","B")))
x
```

```
  A B
X 2 3
Y 4 5
Z 6 7
```

```
#Dimension names can be accessed or changed with two helpful
functions colnames() and rownames():
```

```
colnames(x)
[1] "A" "B"
rownames(x)
[1] "X" "Y" "Z"
colnames(x) <- c("a","b")
colnames(x)
[1] "a" "b"
```

rownames can be changed in similar manner

Matrices

- Another way of creating a matrix is by using functions **`cbind()`** and **`rbind()`** as in column bind and row bind.

```
# Alternative command for creating matrix column-wise and row-wise
```

```
cbind(c(2,3,4),c(5,6,7))

rbind(c(2,3),c(4,5), c(6,7))
```

Arrays

- An array holds multidimensional rectangular data i.e. each row is the same length, and likewise for each column and other dimensions

```
#Create an array with four columns, three rows and two “tables”.
a<-array(1:24,dim=c(3,4,2))
a
```

```
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2

     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24
```

array(data, dim = c(r,c,t))
r = no. of rows
c = no. of columns
t = no. of tables
Note: Although the rows are given as the first dimension, the tables are filled column-wise. So, for arrays, R fills the columns, then the rows, and then the rest

Data Frames

Data frame is a two-dimensional array like structure. It is a list of vectors of equal length. Data frames are the primary data structure in R. To convert any object into a data frame type, use **`as.data.frame()`** function.

Create a data frame from different vectors

```
x<-c(12,23,45)
y<-c(13,21,6)
z<-c("a","b","c")
```

creating vectors x, y, z

```
data<-data.frame(x,y,z)
data
```

- data.frame() function combines them in a table.
- object data is a dataframe containing three vectors x, y, z

```
   x  y z
1 12 13 a
2 23 21 b
3 45  6 c
```

* Data.frames are distinct from matrices because they can include heterogeneous data types among columns/variables.

Data Frames

Let's have a look at the structure of our data frame.

```
str(data)
```

```
'data.frame':    3 obs. of  3 variables:
 $ x: num  12 23 45
 $ y: num  13 21 6
 $ z: Factor w/ 3 levels "a","b","c": 1 2 3
```

- str() shows the structure of an object
- Note : z is a character vector but by default R stores it in the data frame as factor..

*

By default, R always transforms character vectors to factors when creating a data frame with character vectors or converting a character matrix to a data frame. This can cause errors if you are unaware of it. To avoid this, you can specify **stringsAsFactors=FALSE** while creating a dataframe

Lists

List is a data structure having components of mixed data types. To convert any object into a list type, use **`as.list()`** function.

Create a list of 2 vectors and a numeric value

```
n=c(2, 3, 5)
s=c("aa", "bb", "cc", "dd", "ee")
x=list(n, s, 3)
x
```

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

[[2]]
[1] "aa" "bb" "cc" "dd" "ee"

[[3]]
[1] 3
```

list() is used to create lists,

Lists

We retrieve a list slice by enclosing the index of the vector single in a square bracket "[]" operator.

```
x[2]
```

With an index vector, we can retrieve a slice with multiple members.

```
[[1]]
[1] "aa" "bb" "cc" "dd" "ee"
```

```
x[c(2, 3)]
```

```
[[1]]
[1] "aa" "bb" "cc" "dd" "ee"

[[2]]
[1] 3
```

Quick Recap

1. Numeric
2. Integer
3. Character
4. Factor
5. Logical
6. Vector
7. Matrices
8. Array
9. Data Frame
10. List

Data Types

`as.integer()`,
`as.matrix()`,
`as.data.frame()`,
`as.list()`,
`as.logical()`, and
`as.vector()`.

Type Conversion Functions