Data Types

R is a programming language for statistical computing and graphics supported by the R Core Team and the R Foundation for Statistical Computing.

Summary

Data Types

- atomic classes: numeric, logical, integer, complex
- vectors, lists
- factors
- missing values
- data frames
- names

Object

Every thing in R is an object

- Character
- Numeric (Real Number)
- Integer
- Complex
- Logical (Ture/False)

The most basic object is a vector

- Contain multiple object **in the same class**
- List(sequence of objects) can have different class

Vector function (class of object and length)

```
vector()
```

Number

In R called numeric object(double precision real number)

Need to specify `"L"` if want an integer(`1` is numeric object, `1L` is integer)

`inf` is infinity(∞) in R

`NaN` is an Undefined Value(Not a number) or a missing value

Attribute

R objects have attribute

- Names dimnames
- dimensions(Matrices, arrays)
- class
- length
- other user-defined attributes/metadata

Attribute of function can be accessed by

```
attribute()
```

Creating Vectors

the`c()` function can be used to create vectors of objects (c is concatenate, connecting objects together)

```
> x <- c(0.5, 0.6) ## numeric
> x <- c(TURE, FALSE) ## logical
> x <- c(T, F) ## logical
> x <- c("a", "b", "c") ## character
> x <- 9:20 ## integer (create a vector from 9 to 20)
> x <- c(1+0i, 2+4i) ## complex
```

Using vector function

```
> x <- vector("numeric", length = 10)
```

Mixing Object

Create a least common denominator vector

```
> x <- c("1.7", "a") ## character
> x <- c(TURE, 2) ## numeric
> x <- c("a", TURE) ## character
```

coercion will have when it is running

Explicit Coercion

Using `as.*`to coerce from one class to another

```
x <- 0:6
> class(x)
[1] "integer"
> as.numeric(x)
[1] 0 1 2 3 4 5 6
> as.logical(x)
[1] FALSE TURE TURE TURE TURE TURE TURE ## FALSE occur when x=0
> as,character(x)
[1] "0" "1" "2" "3" "4" "5" "6"
```

If it it failed to coerce, the output is `NA`

```
> x <- c("a", "b", "c")
> as.numeric(x)
[1] NA NA NA
```

List

Contain different classes

```
> x <- list(1, "a", TURE, 1+4i)
> x
[[1]] ## index of the list 1 is the element
[1] 1

[[2]]
[1] "a"
```

Matrices

A vector has a dimension attribute, an integer vector of (`nrow ncol`)

```
> x <- matrix(nrow = 2, ncol = 3)
> x
    [.1] [.2] [.3]
[1.] NA   NA   NA
[2.] NA   NA   NA
> dim(m)
[1] 2 3
> attribute(m)
$dim
[1] 2 3
```

Matrices(cont'd)

Matrices are constructed column-wise(from upper left to down)

```
> x<- matrix(1:6, nrow=2, ncol=3)
> x
    [,1] [,2] [,3]
[1,]   1   3    5
[2,]   2   4    6
```

Matrices can also be constructed by adding a dimension attribute

```
> x <- 1:10
> x
[1] 1 2 3 4 5 6 7 8 9 10
> dim(x) <- c(2,5)
## (create a 2x5 matrix)
```

cbind-ing and rbind-ing

Combining 2 vectors to become one matrix

```
> x <- 1:3
> y <- 10:12
> cbind(x, y)
    x y
[1,] 1 10
[2,] 2 11
[3,] 3 12
> rbind(x, y)
    [,1] [,2] [,3]
x   1    2   3
y   10   11  12
```

Factor

Represent categorical data, can be ordered or unordered. Can consider a factor as an integer vector where each integer has a label

Being specially treated by modelling function like `lm()` ,`gim()`

factor with label is better than integers since it is self-describing

```
> x <- factor(c("yes", "yes", "no", "yes", "no"))
> x
[1] yes yes no yes no
Levels: no yes
> table(x) ## frequency of the function
x
no yes
2  3
> unclass(x) ## remove class
[1] 2 2 1 2 1
```

Factors

The order of the levels can be set using `levels` argument to `factor`

```
> x <- factor(c("yes", "yes", "no", "yes", "no")),
            levels = c("yes", "no") ## tells the order
> x
[1] yes yes no yes no
Levels: yes no
```

Missing Value

Missing Values are denoted `NA` or `NaN` for undefined mathematical operations

- `is.na()` is used to test objects if they are `NA`
- `is.nan()` is used to test `NaN`
- `NA` values have a class also, there are integer `NA`, character `NA` .etc
- A `NaN` value is also `NA` but `NA` is not `NaN`

```
> x <- c(1, 2, NA, 10, 3)
> is.na(x)
[1] FALSE FALSE TURE FALSE FALSE ## if have NA, the element is ture
> is.nan(x)
[1] FALSE FALSE FALSE FALSE FALSE ## NaN does not exist
> x <- c(1, 2, NaN, NA, 4)
> is.na(x)
[1] FALSE FALSE TURE TURE FLASE
> is.nan(X)
[1] FALSE FALSE TURE FALSE FALSE ##A NaN value is also NA but NA is not NaN
```

Data Frames

Used to store tabular data

Represent a special type of list where every element of the list has the same length

Each column can be a different type(Element is the column and length is the number of the row)

Unlike matrices, it can store different classes of objects

It also has a special attributes called `row.name` (Every row has its name)

Usually created by `read.table()` or `read.csv()`

Can be converted to a matrix by `data.matrix()`

```
> x <- data.frame(foo = 1:4, bar = c(T, T, F, F))
> x
    foo bar
1   1   TURE
2   2   TURE
3   3   FALSE
4   4   FALSE
> nrow(x)
[1] 4
> ncol(x)
[1] 2
```

Names

R object can also have names, useful for writing readable code and self-describing objects

```
> x <- 1:3
> name(x)
NULL ## did not define the name
> name(x) <- c("foo", "bar", "norf") ## character
> x
 foo bar norf
 1   2   3
> name(x)
[1] "foo" "nar" "norf"
```

List can also have names

```
> x <- list(a = 1, b = 2, c = 3)
> b
$a
[1] 1

$b
[1] 2

$c
[1] 3
```

Also maticies

```
> m <- matrix(1:4, nrow = 2, ncol = 2)
> dimnames(m) <- list(c("a". "b"), c("c", "d"))
> m
    c d
a   1 3
b   2 4
```