



École d'Ingénierie Digitale
et d'Intelligence Artificielle

R Programming

Manipulating data using R



الجامعة الأوروبية بـفاس
EUROMED UNIVERSITY OF FES
UNIVERSITÉ EUROMED DE FÈS

Objects

R manipulates objects. So, when we import a file into R, we get in R an object named data frame. Variables, data, functions, analysis results are stored in objects. There are several types of objects: vectors, factors, etc. The main objects are presented in this part.

The objects are characterized by their name, their content and attributes that will specify the type of data represented by the object.

The name of an object must start with a letter and can include letters, numbers, periods (.), and underscores (_). R distinguishes, for the names of the objects, the upper case letters, that is to say that x and X will name distinct objects.

Objects all have at least two attributes: mode and length. The mode is the type of the elements of an object. There are four main ones: numeric, character, complex, and logical (FALSE or TRUE). To find out the mode or the length of an object, we use the mode () and length () functions respectively.

Objects

Basic objects

The most basic object is a constant, which can be numeric, complex, character, or logical.

We directly assign a value to an object. The object does not need to be declared.

For example, we enter on the console:

```
n = 8
```

We then type `n` to display its value. We obtain the following result:

```
[1] 8
```

The symbol [1] indicates that the display starts at the first element of n.

Objects

Basic objects

We could also have assigned a value to object n using the <- sign (minus sign attached to a square bracket) or ->:

```
n <- 8
```

```
n
```

```
[1] 8
```

```
8 -> n
```

```
n
```

```
[1] 8
```

Objects

Basic objects



Other examples of basic objects:

```
x=1
```

```
x
```

```
[1] 1
```

If you assign a value to an existing object, its previous value is deleted:

```
x = 10
```

```
x
```

```
[1] 10
```

```
y = 10 + 2
```

```
y
```

```
[1] 12
```

Objects

Basic objects

We use ";" to separate different commands on the same line:

```
w = 8; name = "Wikistat"; dicton = "Aide-toi, autrui t'aidera";
```

```
w ; name ; dicton
```

```
[1] 8
```

```
[1] "Wikistat"
```

```
[1] "Aide-toi, autrui t'aidera"
```

Other objects

Objects and their attributes

The following table indicates the possible modes for the vector, factor, array, matrix, data.frame, ts and list objects.

Objet	Modes	Plusieurs modes possibles dans le même objet
vecteur	num, car, comp, log	Non
facteur	num, car	Non
array	num, car, comp, log	Non
matrice	num, car, comp, log	Non
data.frame	num, car, comp, log	Oui
ts	num, car, comp, log	Oui
liste	num, car, comp, log, fonction, expression	Oui

num = numérique, car = caractère, comp = complexe, log = logique

Objects Vectors

*The **vector()** function has two arguments: the mode of the elements that make up the vector and the length of the vector.*

- `a = vector("numeric", 5)`
- `b = vector("character", 5)`
- `c = vector("logical", 5)`
- `d = vector("complex", 5)`

`a = numeric(5)`

`b = character(5)`

`c = logical(5)`

`d = complex(5)`

Objects Vectors

If you type a; b; c; d on the console, the following result is displayed:

```
[1] 0 0 0 0 0
[1] "" "" "" "" ""
[1] FALSE FALSE FALSE FALSE FALSE
[1] 0+0i 0+0i 0+0i 0+0i
```

We can also construct a vector using the function c ()

```
vector = c(5, 7.2, 3.6, 4.9); vector
[1] 5.0 7.2 3.6 4.9
```

Objects

Factors

*The **factor()** function creates nominal categorical variables.*

```
factor( x, levels = sort(unique(x), na.last = TRUE), labels = levels, exclude = NA, ordered = is.ordered(x))
```

- **levels**: specifies what are the possible levels of the factor (by default the unique values of the vector x), i.e. the values that the elements of the factor can take.
- **labels**: defines the names of the levels
- **exclude**: the values of x not to be included in the levels
- **ordered**: logical argument specifying whether the levels of the factor are ordered.

Objects

Matrices

A matrix is a vector that has an additional argument that defines the dimensions of the matrix. All the elements of a matrix must be the same mode.

`matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)`

byrow: indicates whether the values given by data must successively fill the columns (FALSE, by default) or the rows (if TRUE).

dimnames: allows you to give names to rows and columns. You can also give names to the columns or rows of the matrix using the **rownames ()** and **colnames ()** functions.

Objects Matrices

```
matrix(0, 5, 7)
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	0	0	0	0	0	0	0
[2,]	0	0	0	0	0	0	0
[3,]	0	0	0	0	0	0	0
[4,]	0	0	0	0	0	0	0
[5,]	0	0	0	0	0	0	0

```
x = 1:20
```

```
> x
```

```
[1] 1 2 3 4 5 6 7 8 9 10 11 12  
[13] 13 14 15 16 17 18 19 20
```

```
mat1 = matrix(x, 4, 5)
```

```
> mat1
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	1	5	9	13	17
[2,]	2	6	10	14	18
[3,]	3	7	11	15	19
[4,]	4	8	12	16	20

```
mat2 = matrix(x, 4, 5, byrow = TRUE)
```

```
> mat2
```

	[,1]	[,2]	[,3]	[,4]	[,5]
[1,]	1	2	3	4	5
[2,]	6	7	8	9	10
[3,]	11	12	13	14	15
[4,]	16	17	18	19	20

Objects

Matrices

The `paste()` function is useful in a situation where you want to name the rows and/or columns of a matrix. Indeed, the `paste()` function allows you to concatenate objects.

```
nom_var = paste("V", 1:5, sep = "")
```

```
> nom_var
```

```
[1] "V1" "V2" "V3" "V4" "V5"
```

```
nom_ind = paste("I", 1:4, sep = "")
```

```
> nom_ind
```

```
[1] "I1" "I2" "I3" "I4"
```


```
colnames(mat2) = nom_var
```

```
rownames(mat2) = nom_ind
```

```
(or dimnames(mat2) = list(nom_ind, nom_var))
```

Objects

Matrices



```
mat2  
> mat2
```

	V1	V2	V3	V4	V5
I1	1	2	3	4	5
I2	6	7	8	9	10
I3	11	12	13	14	15
I4	16	17	18	19	20

Objects

Data frames

An array of data is implicitly created by the **read.table** function. You can also create a data table with the **data.frame** function.

The elements of a data.frame do not have to have the same mode. Character mode items are considered factors.

All elements of the data.frame must be the same length. Otherwise, the shorter item is "recycled" a whole number of times.

To create a data.frame from a numeric vector and a character vector, we proceed as follows:

Objects

Data frames

To create a data.frame from a numeric vector and a character vector, we proceed as follows:

```
a = c(1, 2, 3) ; a ; mode(a);
```

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

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

```
b = c("a", "b", "c") ; b ; mode(b)
```

```
[1] "a" "b" "c"
```

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

```
df = data.frame(a, b)
```

```
> df
```

	a	b
1	1	a
2	2	b
3	3	c

Objects

Data frames

To create with a single instruction a data.frame with a numeric variable and a character variable:

```
df2 = data.frame(a = 1:6, b = letters[1:6]);
```

```
> df2
```

	a	b
1	1	a
2	2	b
3	3	c
4	4	d
5	5	e
6	6	f

Objects

Data frames

If we juxtapose a vector of length 6 and a vector of length 3, the second vector is duplicated:

```
a = c(1, 2, 3, 4, 5, 6)
```

```
b = c("a", "b", "c")
```

```
df = data.frame(a, b)
```

```
>df
```

	a	b
1	1	a
2	2	b
3	3	c
4	4	d
5	5	e
6	6	f

Objects

Data frames

Therefore, the length of one of the vectors must be a multiple of the length of the other vector:

```
a = c(1, 2, 3, 4, 5)
```

```
b = c("a", "b", "c")
```

```
df = data.frame(a, b)
```

```
Error in data.frame(a, b) : arguments imply differing number of rows: 5, 3
```

Objects

Lists

A list is created in the same way as a data.frame. Like the data.frame, the elements that compose it are not necessarily of the same mode. They are not necessarily the same length, as the following example illustrates:

```
a = c(1, 2, 3, 4, 5)
```

```
b = c("a", "b", "c")
```

```
liste1 = list(a, b)
```

```
> liste1
```

```
[[1]]
```

```
[1] 1 2 3 4 5
```

```
[[2]]
```

```
[1] "a" "b" "c"
```

Objects

Lists

We can name the elements of a list as follows:

```
names(liste1) = c("L1", "L2") ;liste1
```

```
$L1
```

```
[1] 1 2 3 4 5
```

```
$L2
```

```
[1] "a" "b" "c"
```

```
liste2 = list(L1 = a, L2 = b)
```

```
liste2
```

```
$L1
```

```
[1] 1 2 3 4 5
```

```
$L2
```

```
[1] "a" "b" "c"
```

Conversion

Modes conversion

In many practical situations, it is useful to convert the mode from one object to another. Such a conversion will be possible thanks to a function of the form: `as.mode` (`as.numeric`, `as.logical`, `as.character`, ...).

Convert into	Fonction	Rule
numérique	<code>as.numeric</code>	<code>FALSE</code> → 0 <code>TRUE</code> → 1 "1", "2", ... → 1, 2,... "A", ... → NA
logique	<code>as.logical</code>	0 → <code>FALSE</code> autres nombres → <code>TRUE</code> "FALSE" → <code>FALSE</code> "TRUE" → <code>TRUE</code> autres caractères → NA
caractère	<code>as.character</code>	1, 2, ... → "1", "2", ... <code>FALSE</code> → "FALSE" <code>TRUE</code> → "TRUE"

Conversion

Modes conversion

Consider a few simple examples:

Case 1: We want to convert objects from logical mode to numeric mode

```
logic = c(FALSE, FALSE, TRUE, TRUE, FALSE, TRUE)
```

```
conversion_numeric = as.numeric(logic)
```

```
conversion_numeric  
[1] 0 0 1 1 0 1
```

Conversion

Modes conversion

Case 2: We want to convert objects from character mode to numeric mode

```
character = c("1", "2", "3", "A", "/", "T", "%", "-")
```

```
conversion_numeric = as.numeric(character)
```

Warning message:

NAs introduced by coercion

```
conversion_numeric
```

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