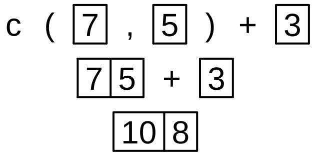
Visual R reference card (Draft)

21 février 2010

Sylvain Loiseau <sylvain.loiseau@unicaen.fr>
Université de Caen–Basse Normandie

1 Graphical conventions

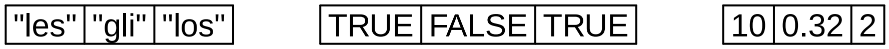
In this document, evaluations of R expressions are represented graphically. For instance, the expression "c(7, 5) + 3", which create a vector and add 3 to its elements, is represented as follow :



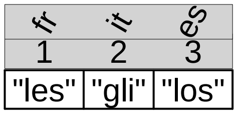
The first line represent the expression you typed in, the last line give the object eventually created, and each intermediary line is a step in the evaluation of the expression.

2 Anatomy of a vector

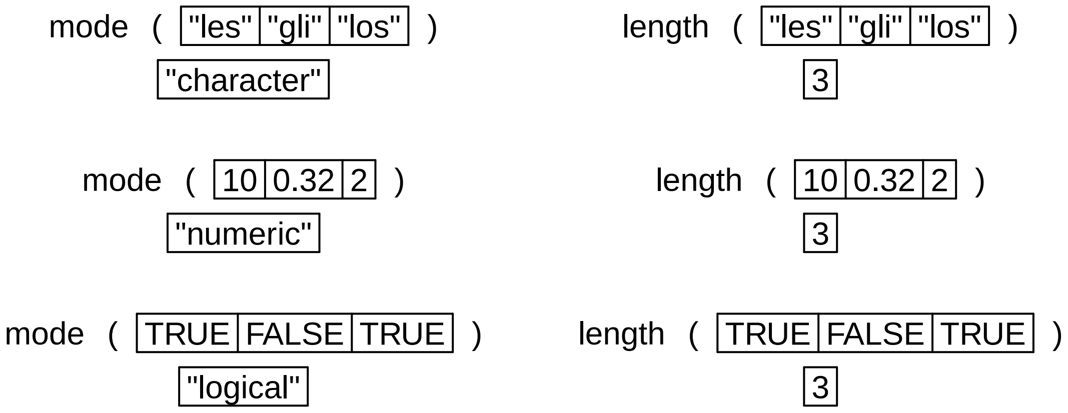
All elements of a vector have a common mode, one of character, logical, and numeric.



All elements of a vector have an index. They may have a name.



All vectors have two important properties : their mode and their length.

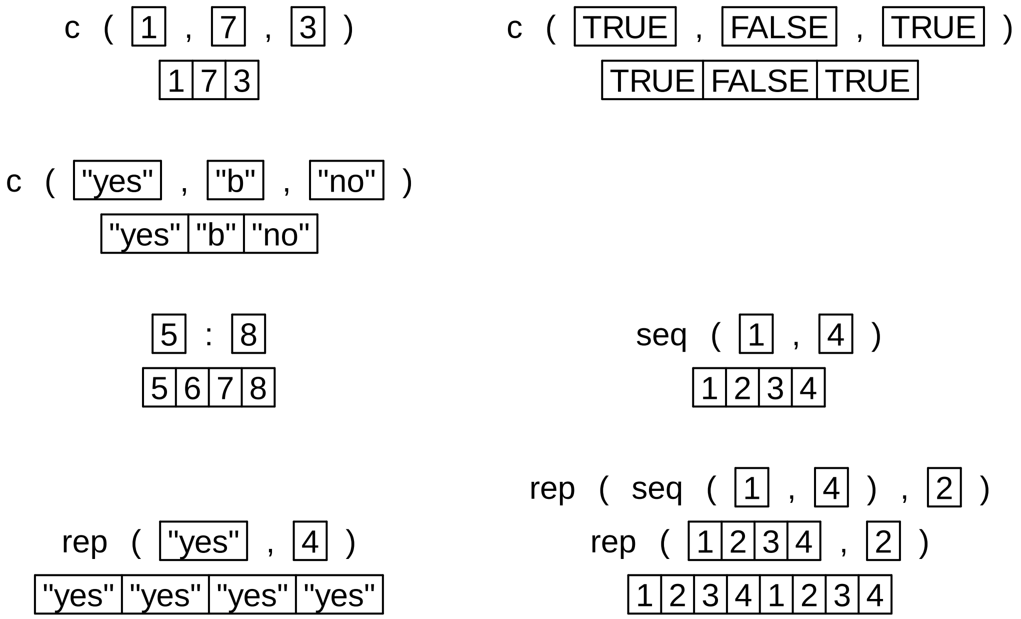


Functions are precisly defined in terms of mode, number and length of vectors they may take as argument, and mode and length of vector they create.

1. length() take one vector of any mode and length and return a vector of numeric vector of length 1.

2. mode() take one vector of any mode and length and return a character vector of length 1.

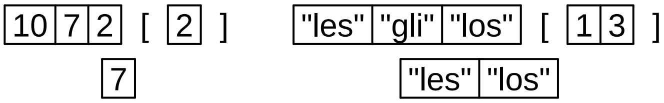
3 Creating a vector



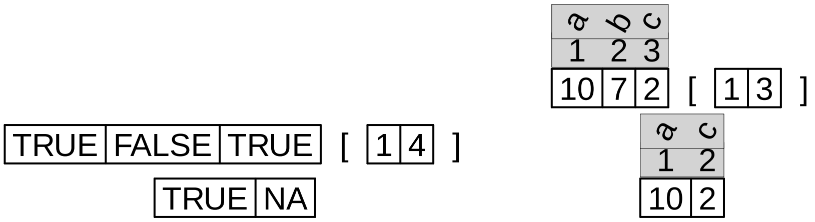
The function c() take any number of vectors of any length and any mode, but all vectors must have the same mode (see below, "Conversion"). It returns a vector of the same mode as the arguments and whose length is the sum of the length of its arguments.

4 Extraction

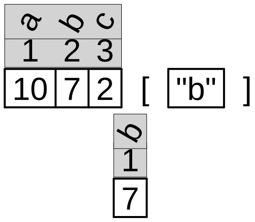
A vector can be created by extracting some elements of a vector.
Elements to be extracted can be addressed using their index.



Index are 1-based. If an index is greater than the number of element in the vector, you get "NA". If the vector has names, their are preserved.



You can also extract with character or logical vector inside the square brackets of the extraction operator. Elements of a character vector are interpreted as the names of the elements to be extracted (elements must have names !).



Logical vectors must have the same length as the vector to be extracted. Elements are extracted if there is a "TRUE" value at the same position in the logical vector. (If the logical vector is shorter, it is recycled : right (see below for recycling))

$$\boxed{10}\boxed{7}\boxed{2}\ [\ \boxed{\text{TRUE}}\boxed{\text{FALSE}}\boxed{\text{TRUE}}\ ] \qquad \boxed{\text{"a"}}\boxed{\text{"b"}}\boxed{\text{"c"}}\boxed{\text{"d"}}\ [\ \boxed{\text{TRUE}}\boxed{\text{FALSE}}\ ]$$

$$\boxed{10}\boxed{2} \qquad \boxed{\text{"a"}}\boxed{\text{"c"}}$$

When extracting, nothing prevent you from reordering elements or extracting several times the same element :

$$\boxed{\text{"a"}}\boxed{\text{"b"}}\boxed{\text{"c"}}\boxed{\text{"d"}}\ [\ \text{c}\ (\ \boxed{1}\ ,\ \boxed{1}\ ,\ \boxed{4}\ ,\ \boxed{2}\ ,\ \boxed{1}\ )\ ]$$

$$\boxed{\text{"a"}}\boxed{\text{"b"}}\boxed{\text{"c"}}\boxed{\text{"d"}}\ [\ \boxed{1}\boxed{1}\boxed{4}\boxed{2}\boxed{1}\ ]$$

$$\boxed{\text{"a"}}\boxed{\text{"a"}}\boxed{\text{"d"}}\boxed{\text{"b"}}\boxed{\text{"a"}}$$

5 Operator

Some numeric operators.

$$\boxed{5}\ +\ \boxed{6} \qquad \boxed{5}\ -\ \boxed{6} \qquad \boxed{5}\ *\ \boxed{6}$$

$$\boxed{11} \qquad \boxed{-1} \qquad \boxed{30}$$

$$\boxed{5}\ /\ \boxed{6} \qquad \boxed{5}\ <\ \boxed{6} \qquad \boxed{5}\ >=\ \boxed{6}$$

$$\boxed{0.833333333333333} \qquad \boxed{\text{TRUE}} \qquad \boxed{\text{FALSE}}$$

Some logical operators.

$$\boxed{5}\ ==\ \boxed{6} \qquad \boxed{5}\ ==\ \boxed{5}$$

$$\boxed{\text{FALSE}} \qquad \boxed{\text{TRUE}}$$

$$\boxed{\text{TRUE}}\ ==\ \boxed{\text{FALSE}} \qquad \boxed{\text{"oui"}}\ ==\ \boxed{\text{"non"}}$$

$$\boxed{\text{FALSE}} \qquad \boxed{\text{FALSE}}$$

6 Vectorization

Operators – as well as many functions – may operate on vector of any length : the operation is performed on pair of elements of equal index.

$$\boxed{4}\boxed{3}\boxed{8}\ +\ \boxed{32}\boxed{3}\boxed{2} \qquad \boxed{4}\boxed{3}\boxed{8}\ ==\ \boxed{32}\boxed{3}\boxed{2}$$

$$\boxed{36}\boxed{6}\boxed{10} \qquad \boxed{\text{FALSE}}\boxed{\text{TRUE}}\boxed{\text{FALSE}}$$

7 Recycling

In a context where vectorization is allowed, you may provide vectors of unequal length. The shorter is duplicated until its length reach the length of the longer. This is called recycling a vector.

$$\text{c}\ (\ \boxed{1}\ ,\ \boxed{2}\ ,\ \boxed{3}\ ,\ \boxed{4}\ )\ +\ \boxed{1}$$

$$\boxed{1}\boxed{2}\boxed{3}\boxed{4}\ +\ \boxed{1}$$

$$\boxed{2}\boxed{3}\boxed{4}\boxed{5}$$

$$\boxed{5}\ :\ \boxed{8}\ >\ \boxed{6}$$

$$\boxed{5}\boxed{6}\boxed{7}\boxed{8}\ >\ \boxed{6}$$

$$\boxed{\text{FALSE}}\boxed{\text{FALSE}}\boxed{\text{TRUE}}\boxed{\text{TRUE}}$$

Be careful :

$$\boxed{\text{"...."}}\boxed{\text{"yes"}}\boxed{\text{"ja"}}\boxed{\text{"si"}}\ ==\ \boxed{\text{"ja"}}\boxed{\text{"yes"}}$$

$$\boxed{\text{FALSE}}\boxed{\text{TRUE}}\boxed{\text{TRUE}}\boxed{\text{FALSE}}$$

8 Some numeric functions

Some functions for numeric vectors.

$$\text{sum}\ (\ \boxed{2}\boxed{1}\boxed{3}\boxed{4}\ )$$

$$\boxed{10}$$

$$\text{mean}\ (\ \boxed{2}\boxed{1}\boxed{3}\boxed{4}\ )$$

$$\boxed{2.5}$$

$$\text{var}\ (\ \boxed{2}\boxed{1}\boxed{3}\boxed{4}\ )$$

$$\boxed{1.66666666666667}$$

$$\text{sd}\ (\ \boxed{2}\boxed{1}\boxed{3}\boxed{4}\ )$$

$$\boxed{1.29099444873581}$$

$$\text{max}\ (\ \boxed{2}\boxed{1}\boxed{3}\boxed{4}\ )$$

$$\boxed{4}$$

$$\text{min}\ (\ \boxed{2}\boxed{1}\boxed{3}\boxed{4}\ )$$

$$\boxed{1}$$

$$\text{range}\ (\ \boxed{2}\boxed{1}\boxed{3}\boxed{4}\ )$$

$$\boxed{1}\boxed{4}$$

$$\text{cumsum}\ (\ \boxed{2}\boxed{1}\boxed{3}\boxed{4}\ )$$

$$\boxed{2}\boxed{3}\boxed{6}\boxed{10}$$

$$\text{cumprod}\ (\ \boxed{2}\boxed{1}\boxed{3}\boxed{4}\ )$$

$$\boxed{2}\boxed{2}\boxed{6}\boxed{24}$$

TODO : table()

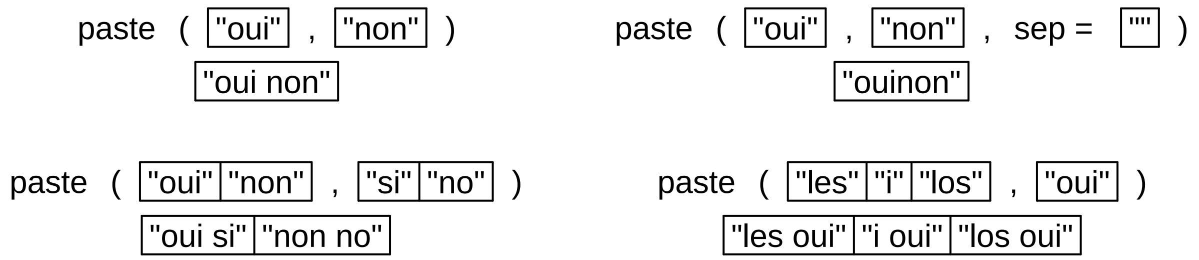
9 Some string functions

nchar() count the number of characters in all strings of a character vector.

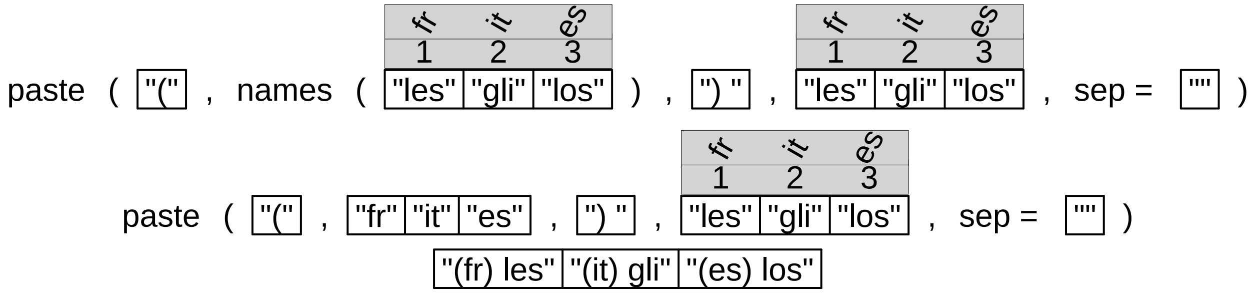
$$\text{nchar}\ (\ \boxed{\text{"les"}}\boxed{\text{"i"}}\boxed{\text{"los"}}\ )$$

$$\boxed{3}\boxed{1}\boxed{3}$$

Recycling and vectorization are useful with paste(), which concatenates characters string at same index in several characters vectors :

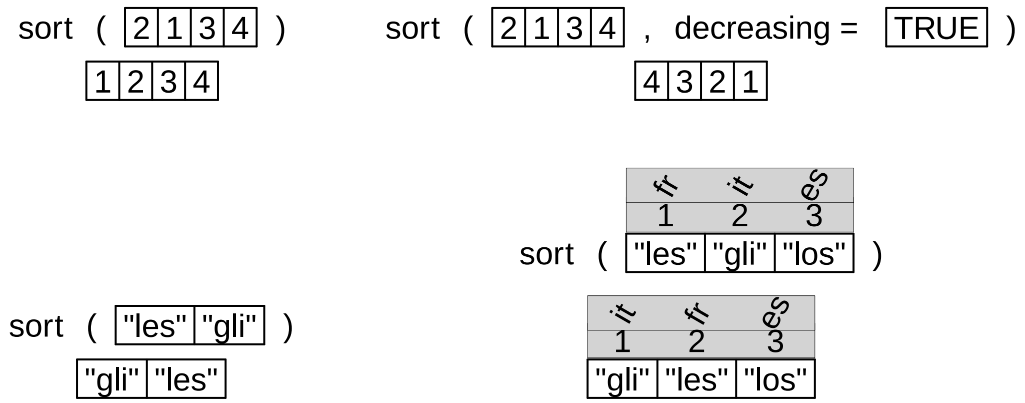

paste ("oui" , "non")
"oui non"
paste ("oui" , "non" , sep = "")
"ouinon"
paste ("oui" "non" , "si" "no")
"oui si" "non no"
paste ("les" "i" "los" , "oui")
"les oui" "i oui" "los oui"

You can paste more than two vectors of characters :

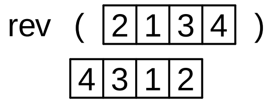

fr it es
1 2 3
paste ("(" , names ("les" "gli" "los") , ") " , "les" "gli" "los" , sep = "")
fr it es
1 2 3
paste ("(" , "fr" "it" "es" , ") " , "les" "gli" "los" , sep = "")
"(fr) les" "(it) gli" "(es) los"

10 Sorting

Numeric and character vectors can be sorted. Names are preserved.

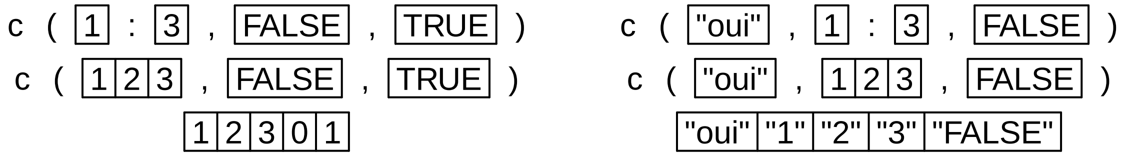


Flipping a vector :

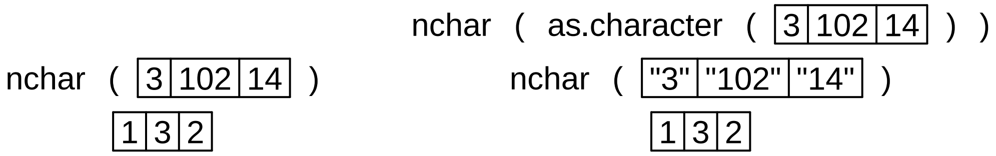


11 Type conversion

c() coerce its arguments to a common mode – all elements of a vector always have a common mode. The character mode always wins. Logical always looses.

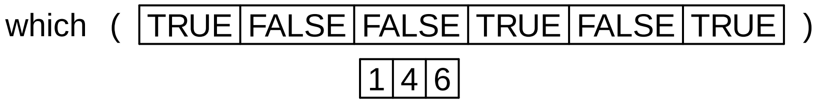

c (1 : 3 , FALSE , TRUE)
c (1 2 3 , FALSE , TRUE)
1 2 3 0 1
c ("oui" , 1 : 3 , FALSE)
c ("oui" , 1 2 3 , FALSE)
"oui" "1" "2" "3" "FALSE"

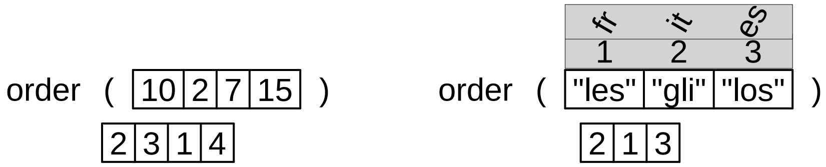
Many functions silently convert their arguments to the requiered mode. For instance, the function nchar() give the number of characters of the elements of a character vector. It makes sense only for characters string : numbers don't have a "number of characters" themselves, but inside a convention of representation as characters strings. If the function nchar() receive a vector of another mode (numerical, logical), the vector is silently converted into a characters vector using as.character() (left). The result is identical to explicitly converting into characters, using as.character() (right).

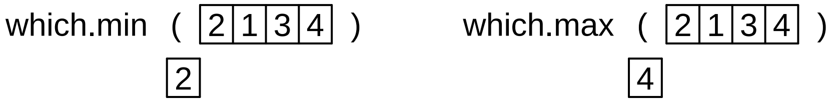


12 Index

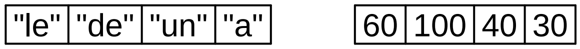
Some useful functions give index rather than the actual values.



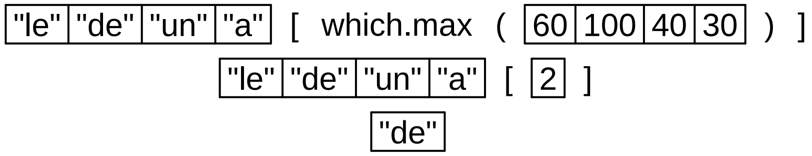




This is particularly usefull in very common situations where two or more vectors are "aligned", or "synchronized". Suppose two vectors : a character vector giving forms in a corpus (left), and a numeric vector giving the total frequency for each form (right) :



Using max(), you may retrieve the maximum frequency from the second vector, but you can't figure out which form have this frequency. Using which.max(), you're still able to extract the corresponding value in the first vector :



Again, suppose you want to sort the row of a matrix according to the value in a column. You cannot use sort, since it give the actual values sorted, not the index of the row sorted. Order is commonly used for reordering data structure :

$$\begin{array}{|c|c|c|}\hline 4&3&5\\\hline 1&2&9\\\hline 3&1&8\\\hline\end{array}\ [\ \text{order}\ (\ \begin{array}{|c|c|c|}\hline 4&3&5\\\hline 1&2&9\\\hline 3&1&8\\\hline\end{array}\ [\ ,\ \boxed{1}\ ]\ )\ ,\ ]$$

$$\begin{array}{|c|c|c|}\hline 4&3&5\\\hline 1&2&9\\\hline 3&1&8\\\hline\end{array}\ [\ \text{order}\ (\ \begin{array}{|c|c|c|}\hline 4&1&3\\\hline\end{array}\ )\ ,\ ]$$

$$\begin{array}{|c|c|c|}\hline 4&3&5\\\hline 1&2&9\\\hline 3&1&8\\\hline\end{array}\ [\ \begin{array}{|c|c|c|}\hline 2&3&1\\\hline\end{array}\ ,\ ]$$

$$\begin{array}{|c|c|c|}\hline 1&2&9\\\hline 3&1&8\\\hline 4&3&5\\\hline\end{array}$$

13 Precedence

Operators have precedence (see ?Syntax).
"seq" takes precedence over "+", "seq" takes precedence over logical operators...

$$\boxed{1}\ :\ \boxed{10}\ +\ \boxed{2}$$

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline 1&2&3&4&5&6&7&8&9&10\\\hline\end{array}\ +\ \boxed{2}$$

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline 3&4&5&6&7&8&9&10&11&12\\\hline\end{array}$$

$$\boxed{5}\ :\ \boxed{8}\ >\ \boxed{6}$$

$$\begin{array}{|c|c|c|c|}\hline 5&6&7&8\\\hline\end{array}\ >\ \boxed{6}$$

$$\begin{array}{|c|c|c|c|}\hline \text{FALSE}&\text{FALSE}&\text{TRUE}&\text{TRUE}\\\hline\end{array}$$

The order in which the operators are written in the code does not matter !

$$\boxed{6}\ <\ \boxed{5}\ :\ \boxed{8}$$

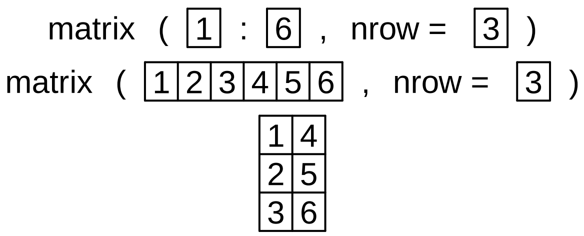
$$\boxed{6}\ <\ \begin{array}{|c|c|c|c|}\hline 5&6&7&8\\\hline\end{array}$$

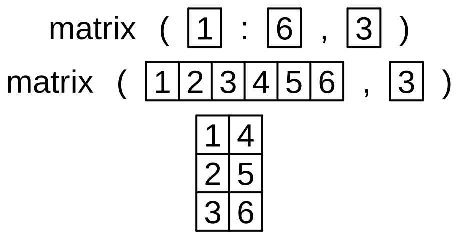
$$\begin{array}{|c|c|c|c|}\hline \text{FALSE}&\text{FALSE}&\text{TRUE}&\text{TRUE}\\\hline\end{array}$$

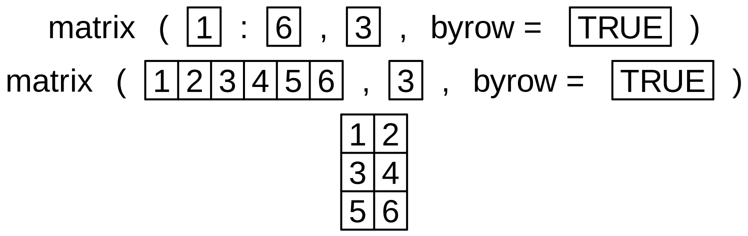
14 Matrix

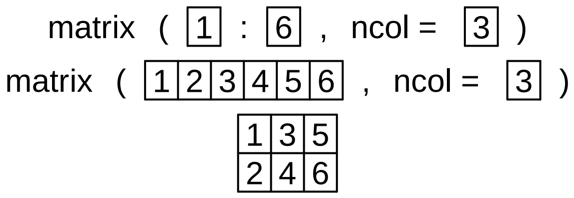
14.1 Creation

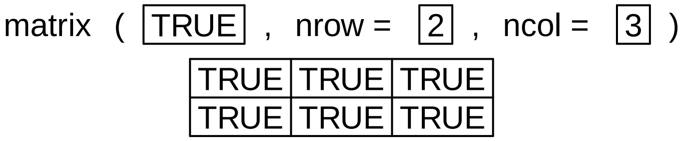
Matrix are created with the matrix() function. It takes three main arguments : a vector (any mode and any length) gives the content, two vectors (numeric and length 1) give the numbers of rows and columns. If only one dimension is given, the second one is deduced from the length of the vector. If both dimensions are given and the vector length do not match the number of cells, the vector is recycled to fill the matrix. The matrix is filled by column ; this behavior may be changed with the option byrow.


matrix (1 : 6 , nrow = 3)
matrix (1 2 3 4 5 6 , nrow = 3)
1 4
2 5
3 6



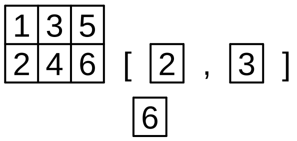

matrix (1 : 6 , 3 , byrow = TRUE)
matrix (1 2 3 4 5 6 , 3 , byrow = TRUE)
1 2
3 4
5 6

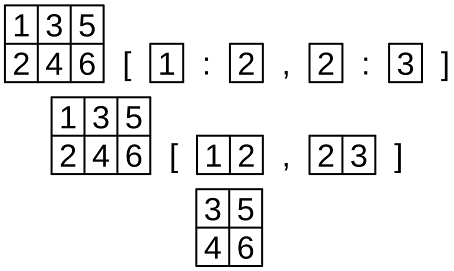

matrix (1 : 6 , ncol = 3)
matrix (1 2 3 4 5 6 , ncol = 3)
1 3 5
2 4 6

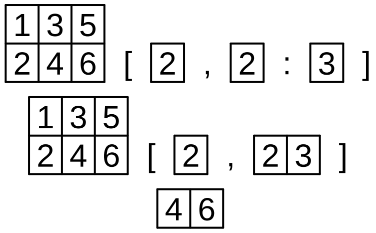


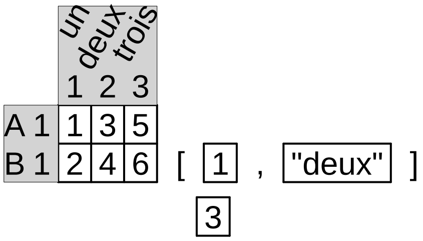
14.2 Extraction with a matrix

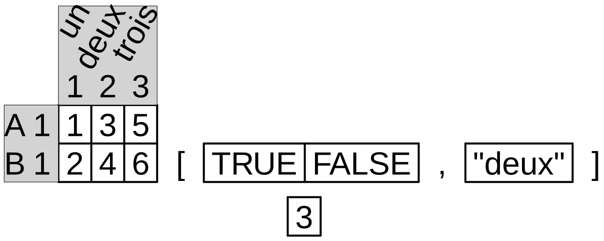
Matrices have two dimensions and you must provide extractors for each of them. You first extract the rows, then the columns.



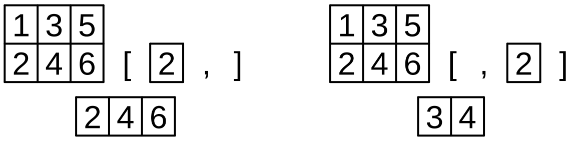




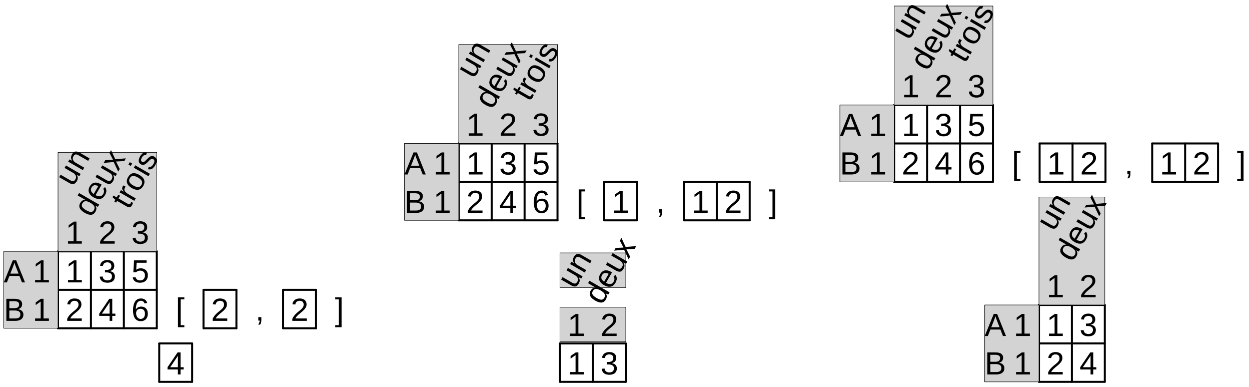




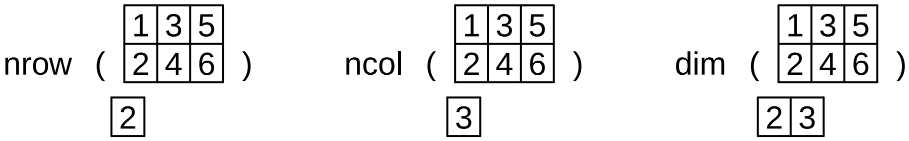
If you leave blank the column slot, all columns are selected (left) ; if you leave blank the row slot, all rows are selected (right).

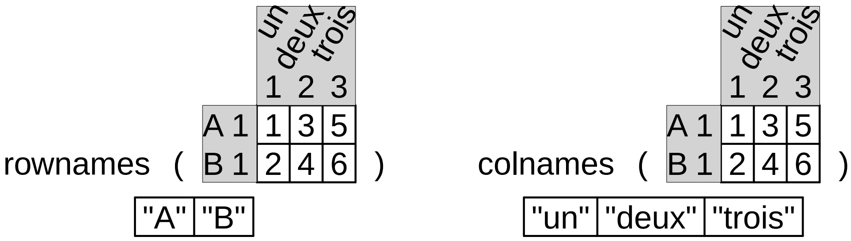


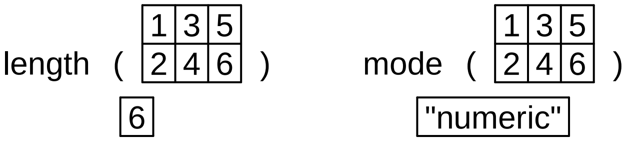
How does extraction in a matrix preserve names ? No name is preserved if you extract a single element ; longuest dimension's names are preserved if you extract a vector of more than one element, both dimensions are preserved if you extract a sub-matrix :



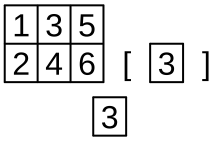
14.3 Properties







A matrix is very similar to a vector : it has a mode and a length. It has also two dimensions and, then, it has two vector of names and it takes two index vectors inside extraction operator. But you can often see a matrix as a vector. For instance, if you use only one index vector inside the extraction operator, it extracts from the underlying, column-filled vector :



14.4 Contingency table

prop.table() compute proportion, given a matrix of frequencies. Proportion are computed either for the whole table (top), by row (middle) or by column (bottom) :

$$\text{prop.table} \;\; (\;\; \begin{array}{|c|c|c|}\hline 1&3&5\\ \hline 2&4&6\\ \hline\end{array} \;\;)$$

0.0476190476190476	0.142857142857143	0.238095238095238
0.0952380952380952	0.19047619047619	0.285714285714286

$$\text{prop.table} \;\; (\;\; \begin{array}{|c|c|c|}\hline 1&3&5\\ \hline 2&4&6\\ \hline\end{array} \;\; , \;\; \text{margin =} \;\; \boxed{1} \;\;)$$

0.111111111111111	0.333333333333333	0.555555555555556
0.166666666666667	0.333333333333333	0.5

$$\text{prop.table} \;\; (\;\; \begin{array}{|c|c|c|}\hline 1&3&5\\ \hline 2&4&6\\ \hline\end{array} \;\; , \;\; \text{margin =} \;\; \boxed{2} \;\;)$$

0.333333333333333	0.428571428571429	0.454545454545455
0.666666666666667	0.571428571428571	0.545454545454545

margin.table() compute margin sum, given a matrix of frequencies. Margin are computed either for the whole table (left), for row (right) or for column (bottom).

$$\text{margin.table} \;\; (\;\; \begin{array}{|c|c|c|}\hline 4&3&5\\ \hline 1&2&9\\ \hline 3&1&8\\ \hline\end{array} \;\;)$$

$$\boxed{36}$$

$$\text{margin.table} \;\; (\;\; \begin{array}{|c|c|c|}\hline 4&3&5\\ \hline 1&2&9\\ \hline 3&1&8\\ \hline\end{array} \;\; , \;\; \text{margin =} \;\; \boxed{1} \;\;)$$

$$\boxed{12} \;\; \boxed{12} \;\; \boxed{12}$$

$$\text{margin.table} \;\; (\;\; \begin{array}{|c|c|c|}\hline 4&3&5\\ \hline 1&2&9\\ \hline 3&1&8\\ \hline\end{array} \;\; , \;\; \text{margin =} \;\; \boxed{2} \;\;)$$

$$\boxed{8} \;\; \boxed{6} \;\; \boxed{22}$$

This objects with dashed lines are "lists" : see below.

14.5 Summing a matrix

$$\text{sum} \;\; (\;\; \begin{array}{|c|c|c|}\hline 1&3&5\\ \hline 2&4&6\\ \hline\end{array} \;\;)$$

$$\boxed{21}$$

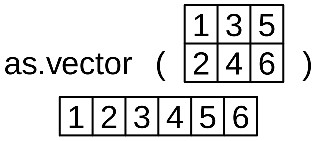
$$\text{rowSums} \;\; (\;\; \begin{array}{|c|c|c|}\hline 1&3&5\\ \hline 2&4&6\\ \hline\end{array} \;\;)$$

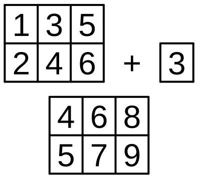
$$\begin{array}{|c|c|}\hline 9&12\\ \hline\end{array}$$

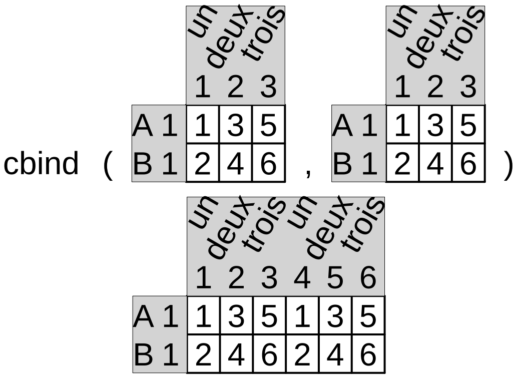
$$\text{colSums} \;\; (\;\; \begin{array}{|c|c|c|}\hline 1&3&5\\ \hline 2&4&6\\ \hline\end{array} \;\;)$$

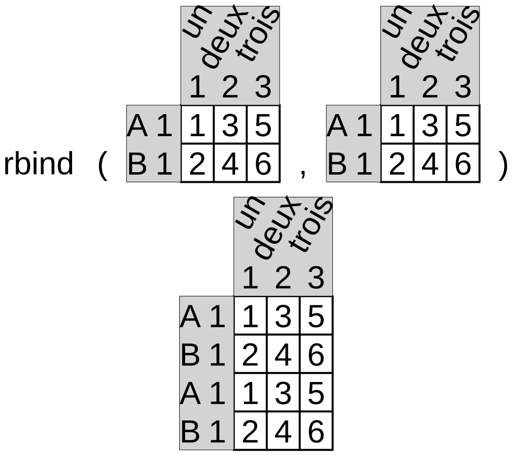
$$\begin{array}{|c|c|c|}\hline 3&7&11\\ \hline\end{array}$$

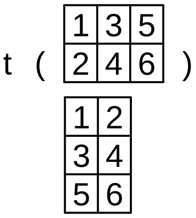
14.6 Changing









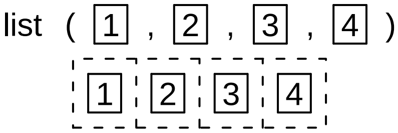


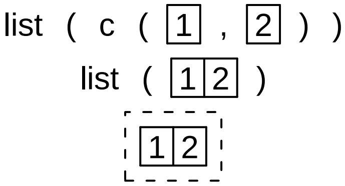
15 list

15.1 Creation, anatomy

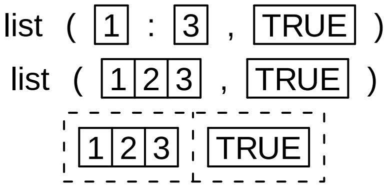
Creating a list by enumerating its components.

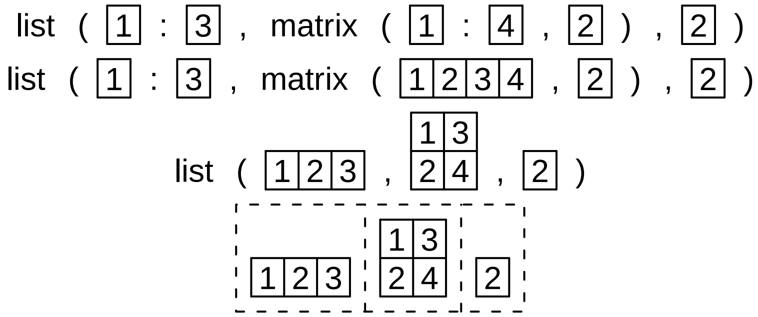
A list of length 4 : contains 4 vectors, each of length 1 (left) ; a list of length 1 : contains 1 vector of length 2 (right).



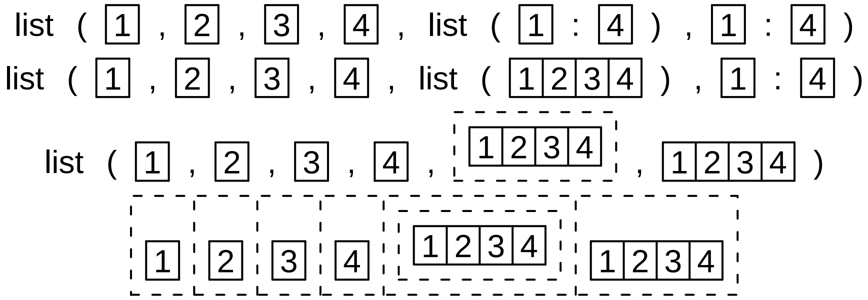


List can contain objects of different mode (left) ; it can contain objects of different dimensions (right).

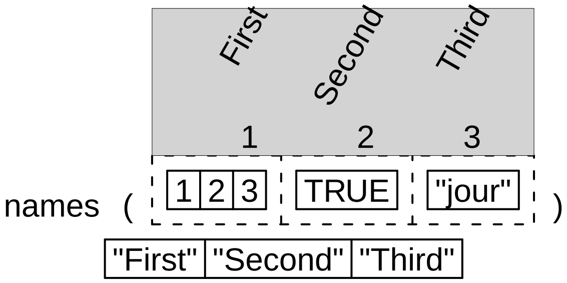


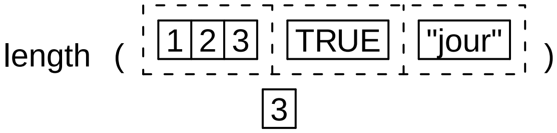

list (1 : 3 , matrix (1 : 4 , 2) , 2)
list (1 : 3 , matrix (1 2 3 4 , 2) , 2)
list (1 2 3 , 1 3 / 2 4 , 2)
1 2 3 | 1 3 / 2 4 | 2

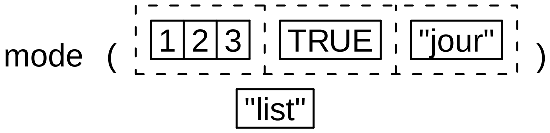
A list can be recursive : a component may be a list.


list (1 , 2 , 3 , 4 , list (1 : 4) , 1 : 4)
list (1 , 2 , 3 , 4 , list (1 2 3 4) , 1 : 4)
list (1 , 2 , 3 , 4 , 1 2 3 4 , 1 2 3 4)
1 | 2 | 3 | 4 | 1 2 3 4 | 1 2 3 4

15.2 Basic functions

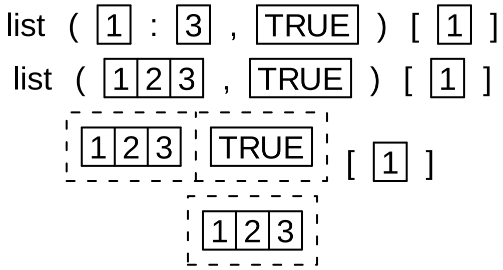


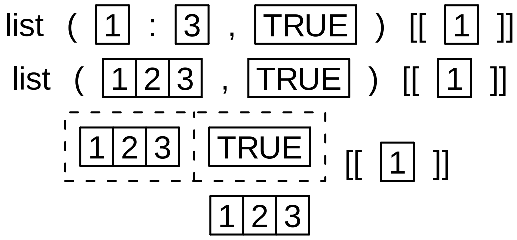




15.3 Extraction

Extracting in a list. The next two figures show the difference beween [and [[operator on list : the first create a sublist (it extracts elements, exactly as it extracts elements from a vector), while the second is completely different : it give the content of one (and only one) element of a list.

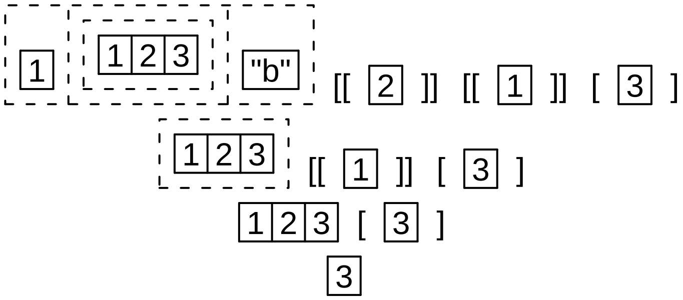




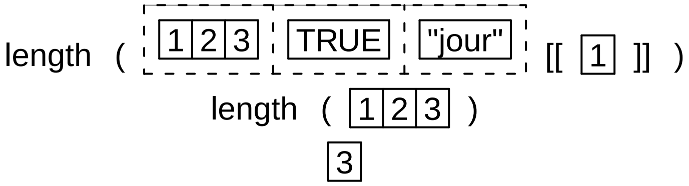
You can use vector of any length within the single-square bracket, while you can address only one element within the double-square-bracket operator, and then use only vector of length 1.

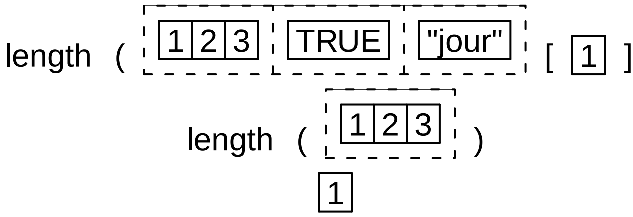
Since a list is a recursive data structure (may contain list), you can use several successive bracket operators in order to go down to the element you're interested in.

With the single-square-bracket operator you cannot walk down though the data structure.

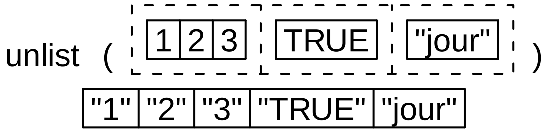


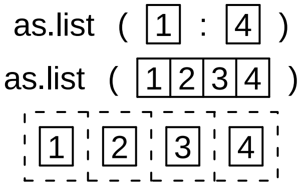
Be sure to understand the difference between "lengt(l[1])" and length(l[[1]])" :





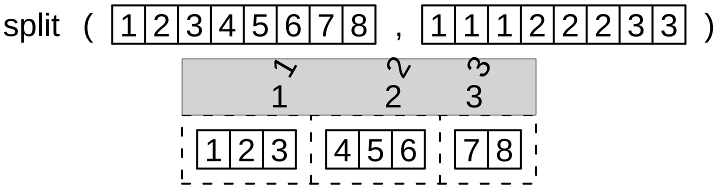
15.4 List and vector





15.5 List for expressing complex data structure

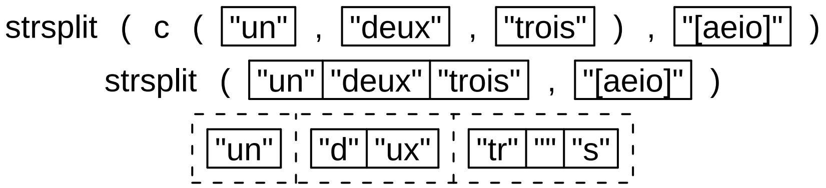
Grouping elements of a vector using a the level of a factors (see Factor) :



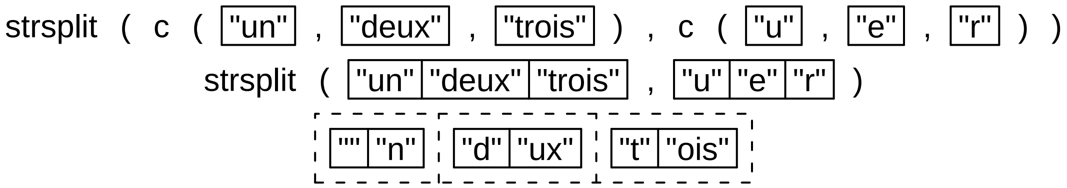
See also for instance strsplit() below.

16 Regexp

16.1 Split strings : strplit()

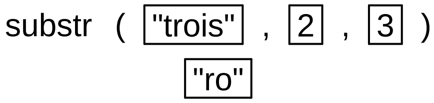


```
strsplit ( c ( "un" , "deux" , "trois" ) , "[aeio]" )
        strsplit ( "un" "deux" "trois" , "[aeio]" )
                  "un"  "d" "ux"  "tr" "" "s"
```

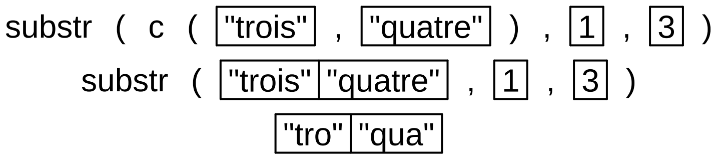


```
strsplit ( c ( "un" , "deux" , "trois" ) , c ( "u" , "e" , "r" ) )
                strsplit ( "un" "deux" "trois" , "u" "e" "r" )
                          "" "n"  "d" "ux"  "t" "ois"
```

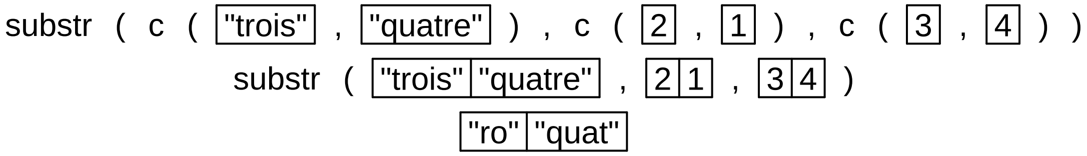
16.2 Extract sub strings : substr()



```
substr ( "trois" , 2 , 3 )
               "ro"
```

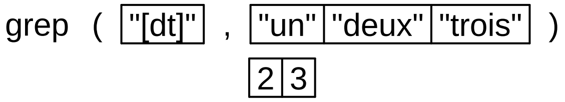


```
substr ( c ( "trois" , "quatre" ) , 1 , 3 )
      substr ( "trois" "quatre" , 1 , 3 )
                 "tro" "qua"
```

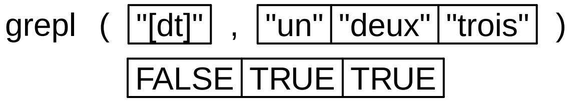


```
substr ( c ( "trois" , "quatre" ) , c ( 2 , 1 ) , c ( 3 , 4 ) )
                substr ( "trois" "quatre" , 2 1 , 3 4 )
                           "ro" "quat"
```

16.3 Searching elements of a character vector with regexp : grep()

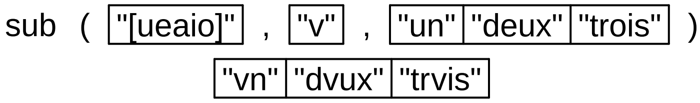


```
grep ( "[dt]" , "un" "deux" "trois" )
                2 3
```

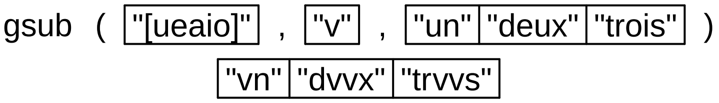


```
grepl ( "[dt]" , "un" "deux" "trois" )
        FALSE TRUE TRUE
```

16.4 Substitution : sub()



```
sub ( "[ueaio]" , "v" , "un" "deux" "trois" )
           "vn" "dvux" "trvis"
```

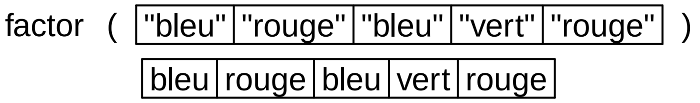


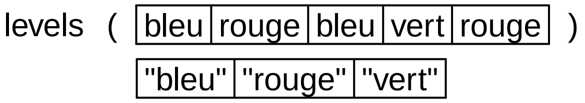
16.5 Searching substring in elements of character vector : regular expression

TODO

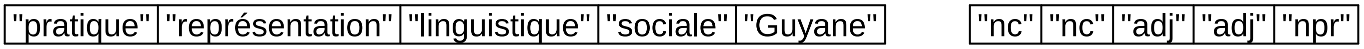
17 Factor

A factor represent a nominal random variable. It looks like a character vector, and may be created using a character vector (left). In this document, factors are represented without quotes around the values. The different values in the factor (the different modality in the random variable) may be retrieved using levels() (right).

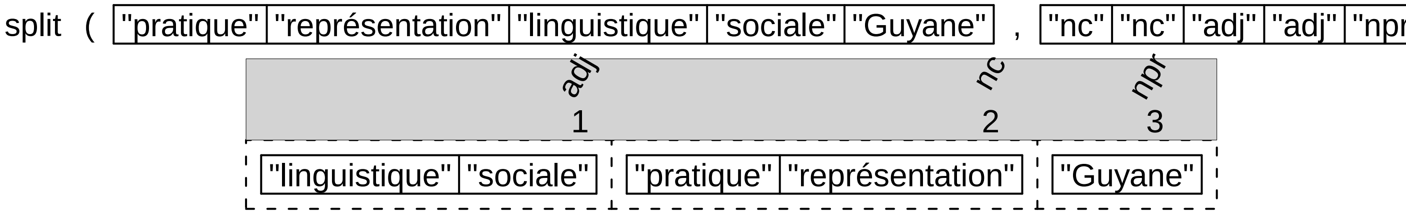

factor ("bleu" "rouge" "bleu" "vert" "rouge")
bleu rouge bleu vert rouge



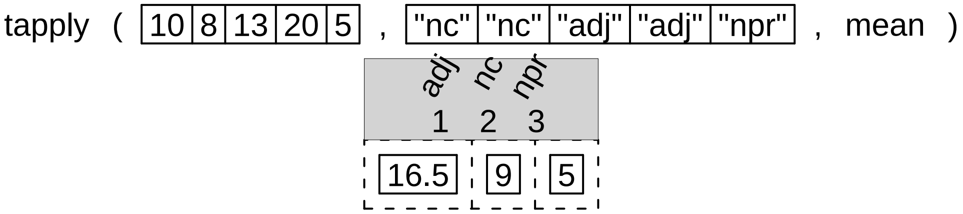
A factor is usefull for *grouping* elements. Suppose two vectors, one giving forms, and the other giving part of speech. You want to group the forms according to part of speech.


"pratique" "représentation" "linguistique" "sociale" "Guyane"
"nc" "nc" "adj" "adj" "npr"

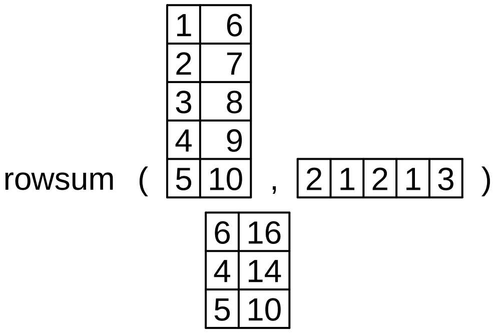
The function split() groups elements, given two arguments : a vector (the elements to be grouped), and a factor (giving the group of each element). split() create a list, each element of the list corresponding to a group, the name of the list element corresponding to the group name.


split ("pratique" "représentation" "linguistique" "sociale" "Guyane" , "nc" "nc" "adj" "adj" "npr
adj
1
nc
2
npr
3
"linguistique" "sociale"
"pratique" "représentation"
"Guyane"

You may give vector of any mode as second argument to split() : split() will call as.factor() on this argument. tapply() do the same grouping, and then apply a function (it's third argument) to each group :


tapply (10 8 13 20 5 , "nc" "nc" "adj" "adj" "npr" , mean)
adj
nc
npr
1 2 3
16.5
9
5

rowsum() performs a colSums on each group of rows given by the second argument.



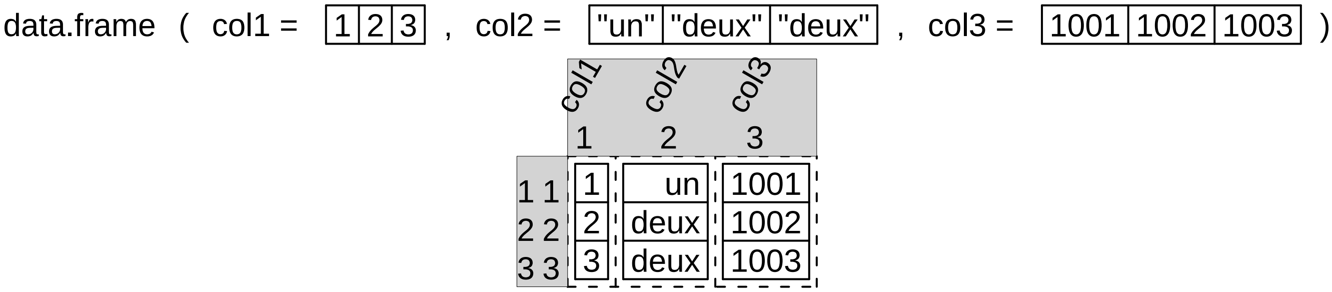
There are numerous other functions using factor (or converting vector into factor) allowing for grouping (by(), aggregate(), etc.)

18 Data Frame

A data frame is a data structure for representing statistical information about a group of individuals. For each individual, you may have numerical random variable, categorial random variable, ie. different modes (numeric, character, factor).

In a data frame, each row represent an individual, and each column represent a random variable. This is like a matrix, except that the columns may have different mode. This is like a list, since it may mix vectors of different modes, but all vectors must have the same length.

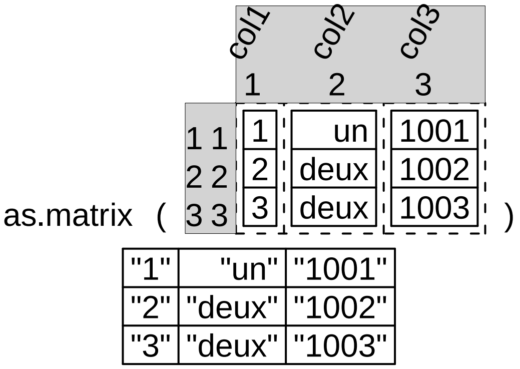
From an internal representation, a data frame is a list of vectors. Thus, data frame may be created with the function data.frame() by enumerating its column-vectors. It is represented here with dotted lines, like a list, and solid lines around vector-column :

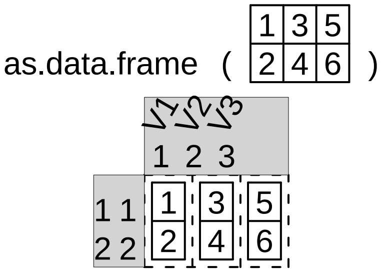

data.frame (col1 = 1 2 3 , col2 = "un" "deux" "deux" , col3 = 1001 1002 1003)
col1 col2 col3
1 2 3
1 1 1 un 1001
2 2 2 deux 1002
3 3 3 deux 1003

Like a matrix and unlike a list, a data frame may have rownames. In fact, a data frame has *always* row and column names (while matrix may have no row or column names). You can see from the previous exemple that automatic default row names have been added : a character representation of the index number of the row. Default colum names are created as well for unnamed column. rownames() and colnames() may be used with data frame as with matrix.

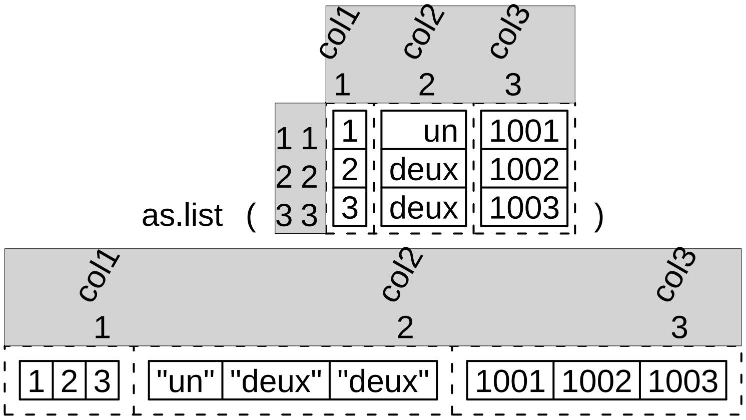
18.1 Converting

You can also create a data frame from a matrix, or create a matrix from a data frame (when you convert data frame into matrix, a mode compatible with all column is used : the character mode below) :

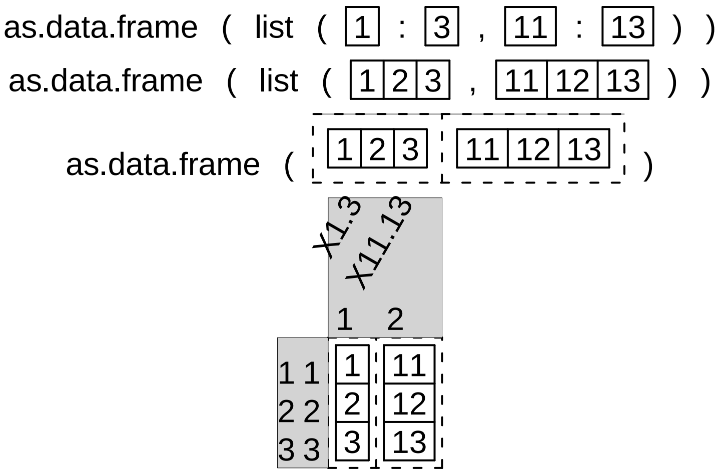




as.data.frame create default row and column names : data frame cannot be without row and column names. Row and column names are lost with as.matrix. Column names are kept with as.list() :



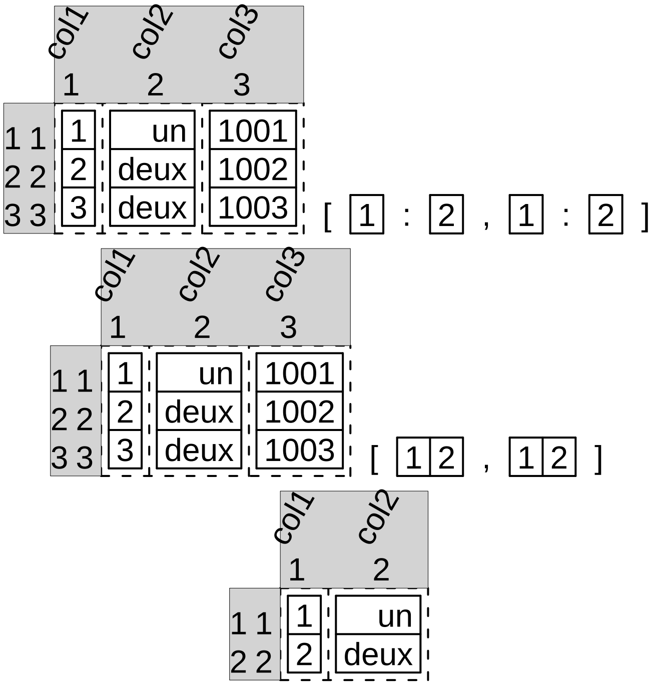
You may create a data frame from a list only if all list component have the same length :


as.data.frame (list (1 : 3 , 11 : 13))
as.data.frame (list (1 2 3 , 11 12 13))
as.data.frame (1 2 3 11 12 13)
X1.3
X11.13
1
2
1 1
2 2
3 3
1
2
3
11
12
13

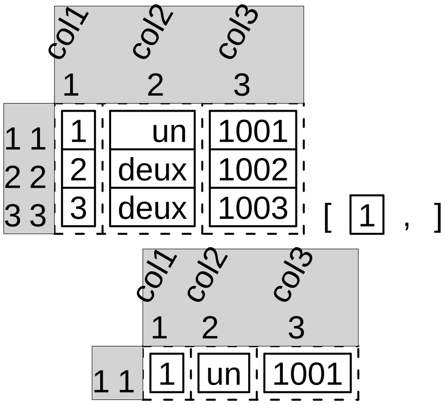
Many functions expecting a matrix will accept a data frame, silently converting it into a matrix. Similarly, many functions expecting a data frame will accept a matrix and silently convert it into a data frame.

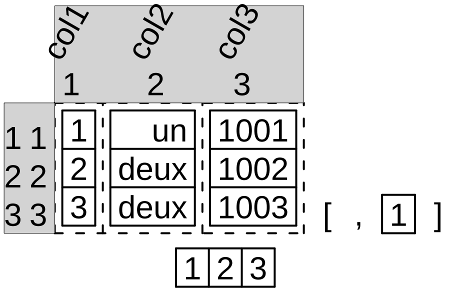
18.2 Extraction

1/ with simple square bracket operator, a data frame is seen as matrix : two dimensions must be provided inside the square brackets

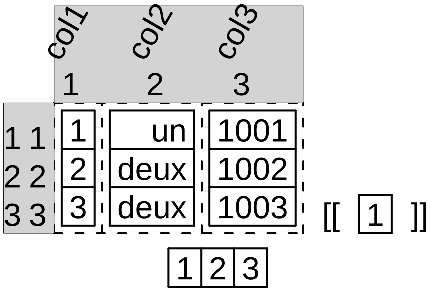


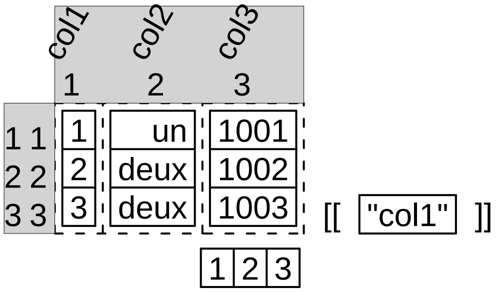
However, note that row-extraction (left) and column-extraction (right) do not give the same data structure. It is a data frame (left) or a vector (right). Furthermore, in column extraction, the row name are lost (right) :



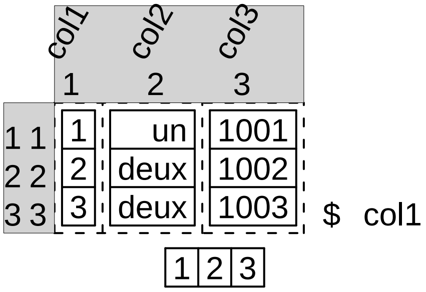


2/ with double square bracket operator, a data.frame behave like a list : one dimension must be provided.





The "$" operator may be used like in a list

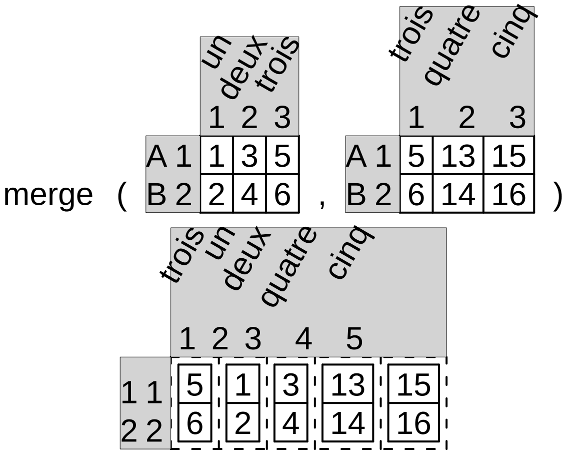


18.3 Subseting

To be done

18.4 Merging

Merge() use the columns with same name for combining two matrices or data frame :



The result is always a data frame. Columns to be used for combining may be given explicitly