Visual R reference card (Draft)

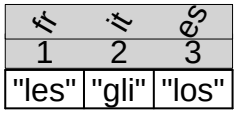
20 février 2010

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

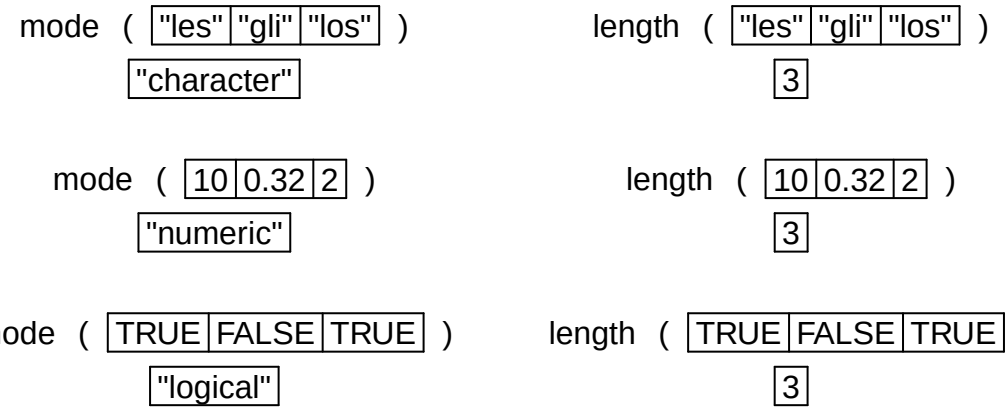
1 Anatomy of a vector

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.



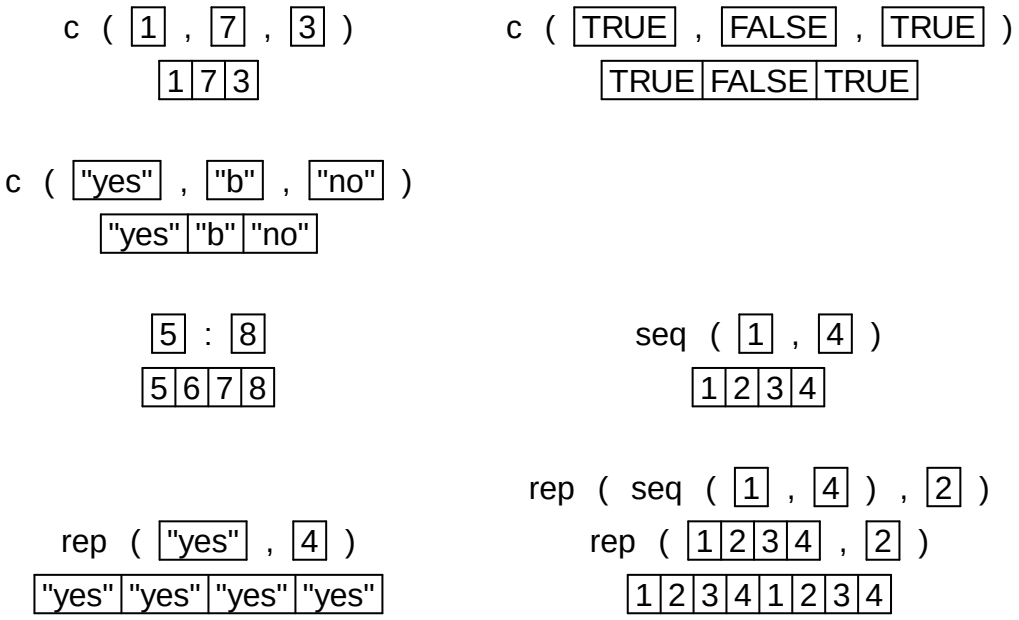
All vectors have two important properties : their mode and their length.



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.

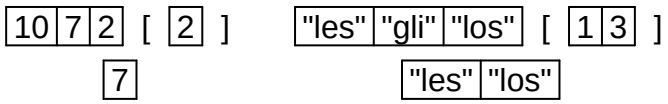
2 Creating a vector



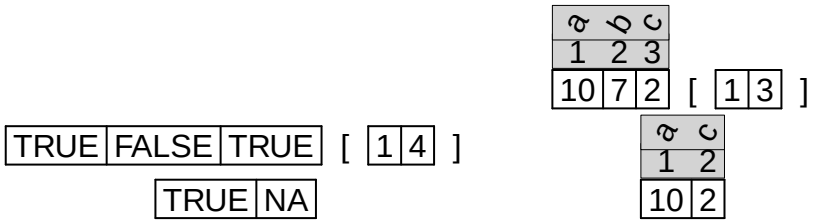
The function c() take any number of vectors of any length and any mode, but all vectors must have the same mode (see below, "Conversion"). It returns a vector of the same mode as the arguments and whose length is the sum of the length of its arguments.

3 Extraction

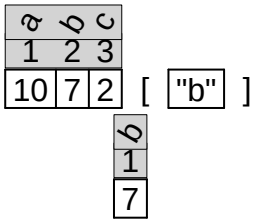
A vector can be created by extracting some elements of a vector.
Elements to be extracted can be addressed using their index.



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.

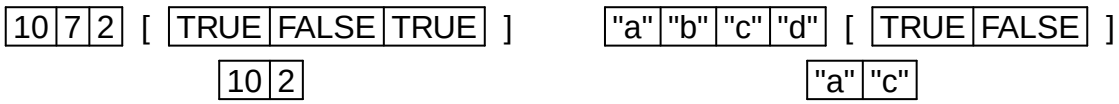


You can also extract with character or logical vector inside the square brackets of the extraction operator. Elements of a character vector are interpreted as the names of the elements to be extracted (elements must have names!).

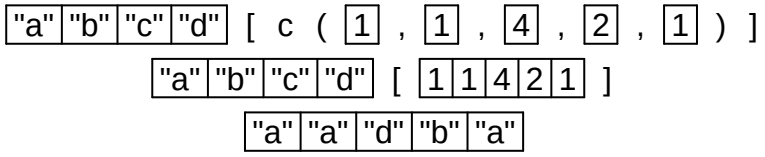


Logical vectors must have the same length as the vector to be extracted. Elements are extracted if there is a "TRUE" value at the same position in the logical vector. (If the logical vector is shorter, it is recycled : right (see

below for recycling))

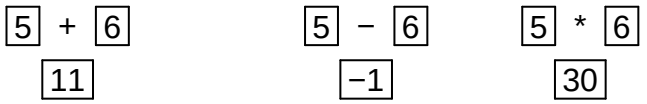


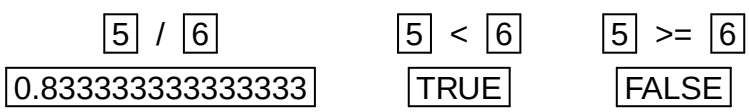
When extracting, nothing prevent you from reordering elements or extracting several times the same element :



4 Operator

Some numeric operators.

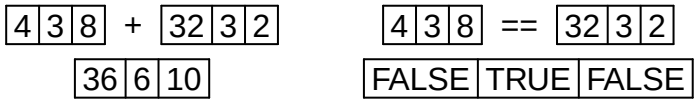




Some logical operators.

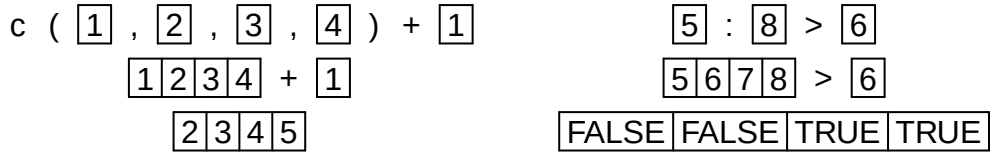
5 Vectorization

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



6 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.



Be careful :

"...."	"yes"	"ja"	"si"	==	"ja"	"yes"
FALSE	TRUE	TRUE	FALSE			

7 Some numerical functions

Some functions for numeric vectors.

sum (c (2 , 1 , 3 , 4))

sum (2 1 3 4)

10

mean (2 1 3 4)

2.5

range (2 1 3 4)

1 4

rev (2 1 3 4)

4 3 1 2

max (2 1 3 4)

4

min (2 1 3 4)

1

cumsum (2 1 3 4)

2 3 6 10

TODO : table()

8 Some string functions

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

nchar ("les" "i" "los")

3 1 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"

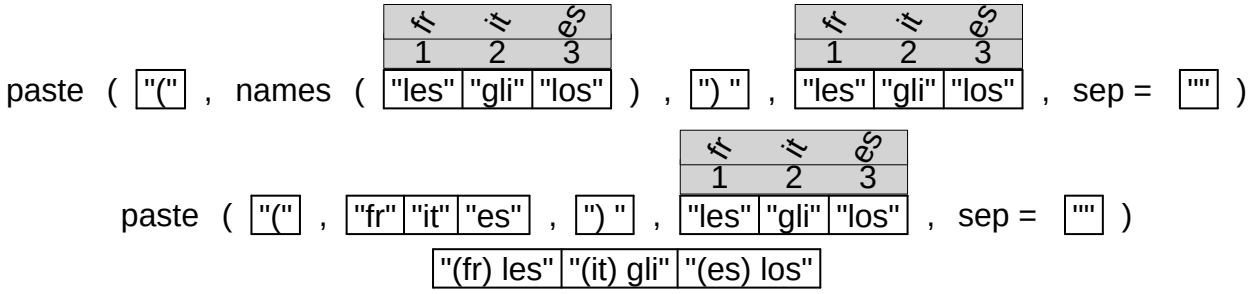
paste ("oui" "non" , "si" "no")

"oui si" "non no"

paste ("les" "i" "los" , "oui")

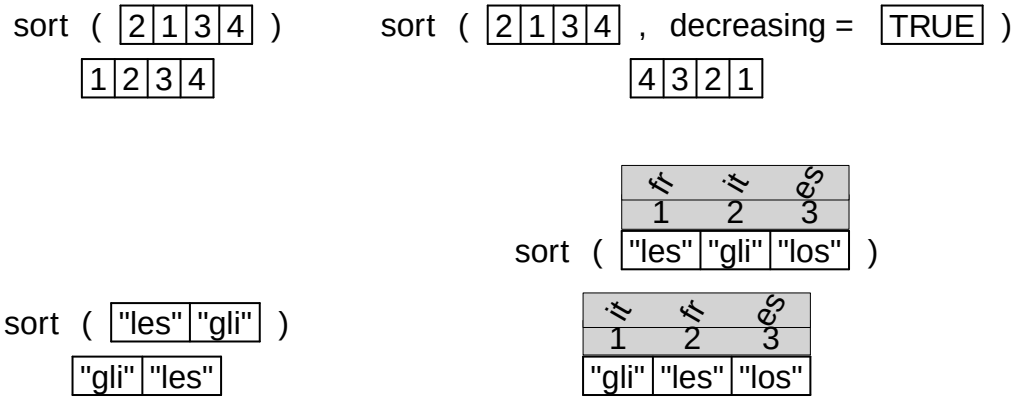
"les oui" "i oui" "los oui"

You can paste more than two vectors of characters :



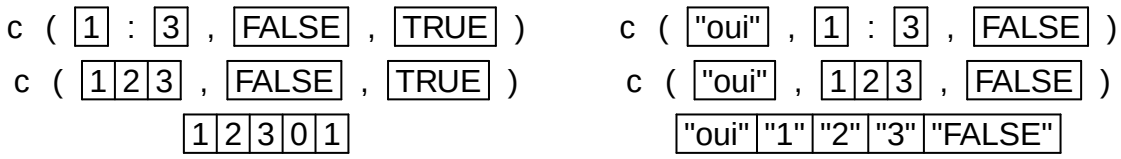
9 Sorting

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

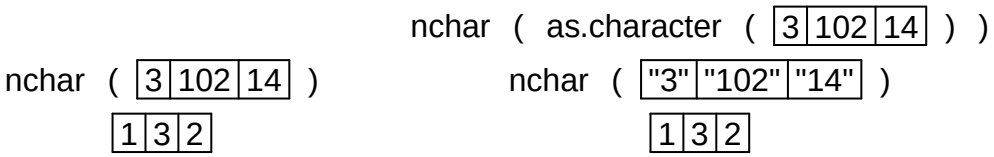


10 Type conversion

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

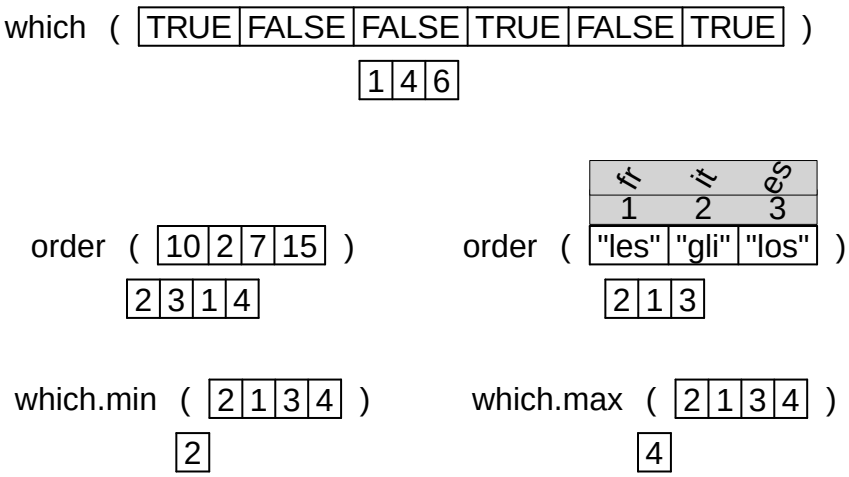


Many functions silently convert their arguments to the requiered type. For instance, the function nchar() give the number of characters of the elements of a character vector. It makes sense only for characters string : numbers don't have a "number of characters" themselves, but inside a convention of representation as characters strings. If the function nchar() receive a vector of another type (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).



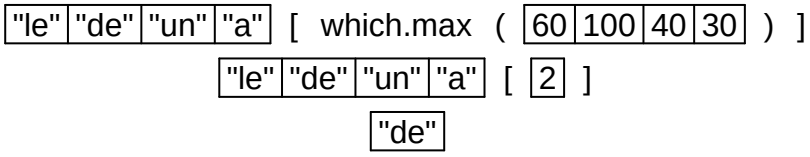
11 Index

Some useful functions give index rather than the actual values.

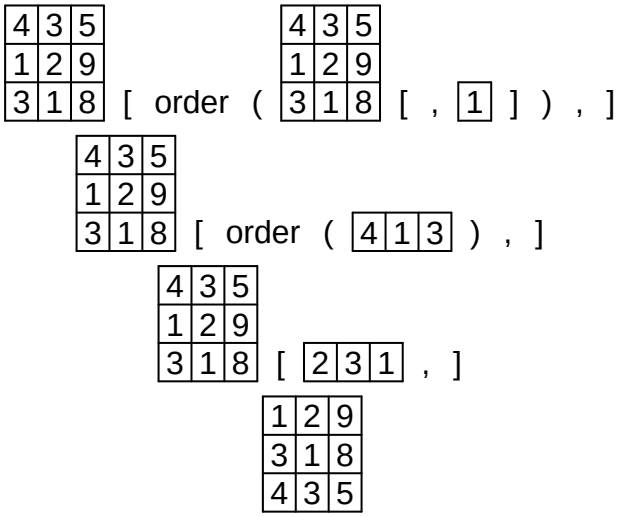


This is particularly usefull in very common situations where two or more vectors are "aligned", or "synchronized". Suppose two vectors : a character vector giving forms in a corpus (left), and a numeric vector giving the total frequency for each form (right) :

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 :



12 Precedence

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

```
1 : 10 + 2
1 2 3 4 5 6 7 8 9 10 + 2
3 4 5 6 7 8 9 10 11 12
```

```
5 : 8 > 6
5 6 7 8 > 6
FALSE FALSE TRUE TRUE
```

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

```
6 < 5 : 8
6 < 5 6 7 8
FALSE FALSE TRUE TRUE
```

13 Matrix

13.1 Creation

Matrix are created with the matrix() function. It takes three main arguments : a vector (any type 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 )
matrix ( 1 2 3 4 5 6 , 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 )
```

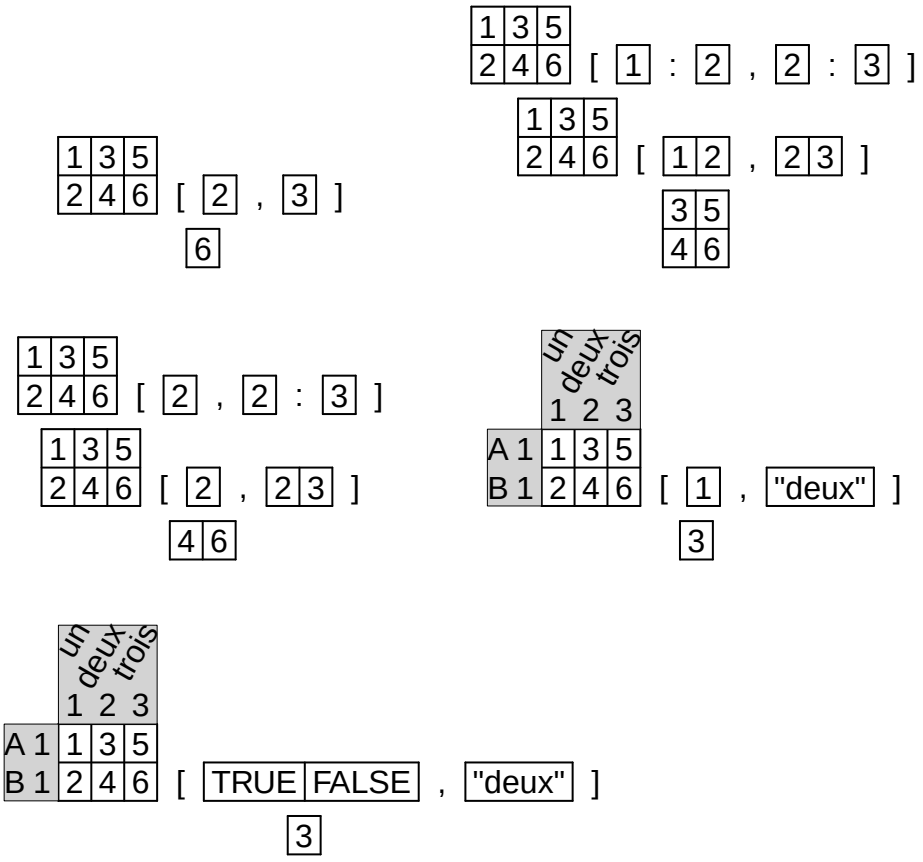
1	3	5
2	4	6

```
matrix ( TRUE , nrow = 2 , ncol = 3 )
```

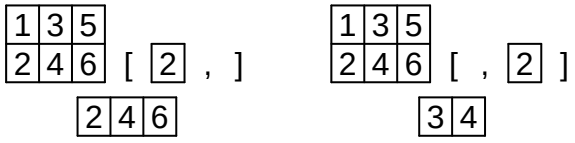
TRUE	TRUE	TRUE
TRUE	TRUE	TRUE

13.2 Extraction with a matrix

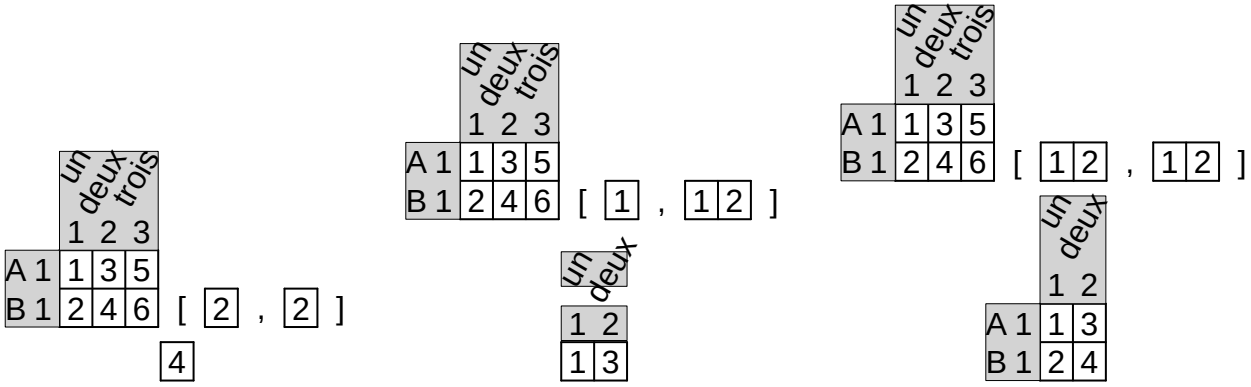
Matrices have two dimensions and you must provide extractors for each of them. You first extract the rows, then the columns.



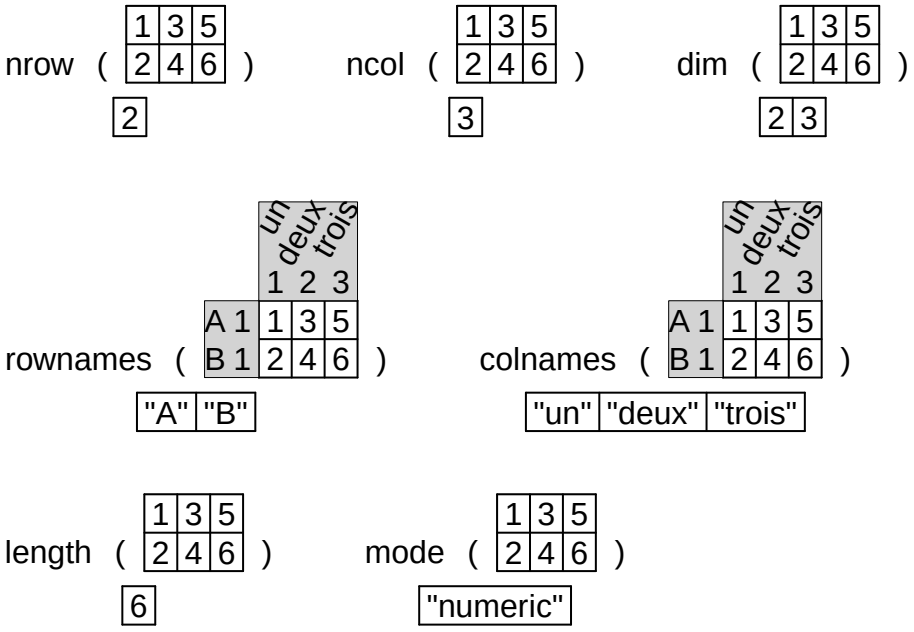
If you leave blank the column slot, all columns are selected (left) ; if you leave blank the row slot, all rows are selected (right).



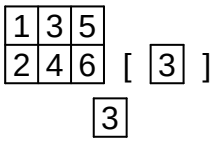
How does extraction in a matrix preserve names ? No name is preserved if you extract a single element ; longuest dimension's names are preserved if you extract a vector of more than one element, both dimensions are preserved if you extract a sub-matrix :



13.3 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 :



13.4 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) :

prop.table (1 3 5 / 2 4 6)

0.0476190476190476	0.142857142857143	0.238095238095238
0.0952380952380952	0.19047619047619	0.285714285714286

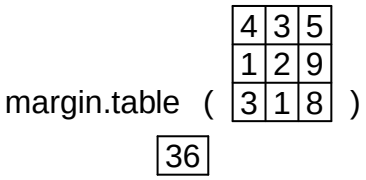
prop.table (1 3 5 / 2 4 6 , margin = 1)

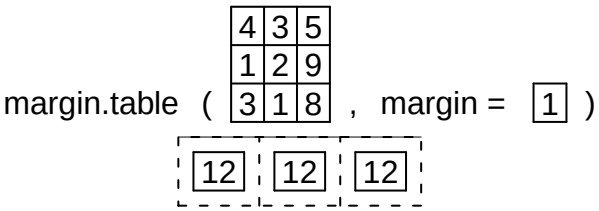
0.111111111111111	0.333333333333333	0.555555555555556
0.166666666666667	0.333333333333333	0.5

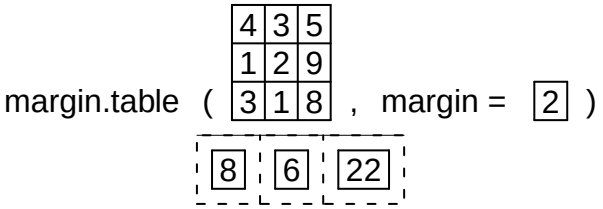
prop.table (1 3 5 / 2 4 6 , margin = 2)

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).

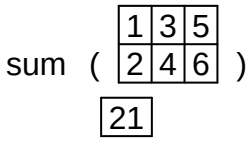


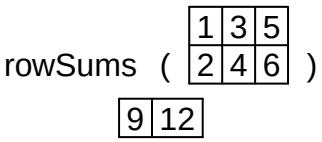


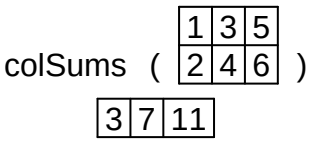


The result with dashed lines are "list" : see below.

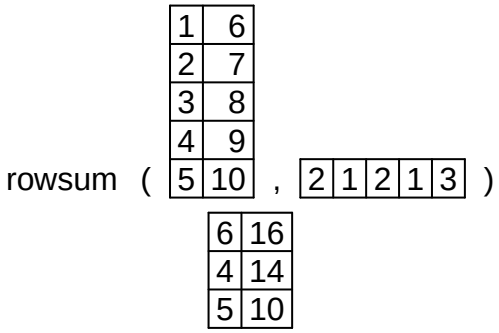
13.5 Summing a matrix



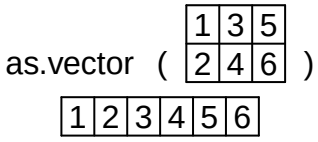


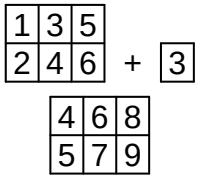


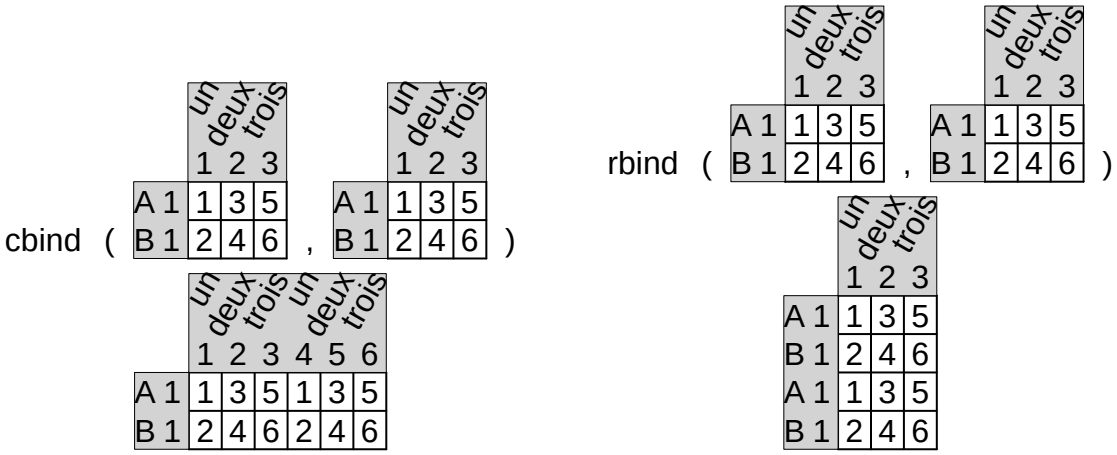
rowsum() performs a colSums on each group of rows given by the second argument.



13.6 Changing





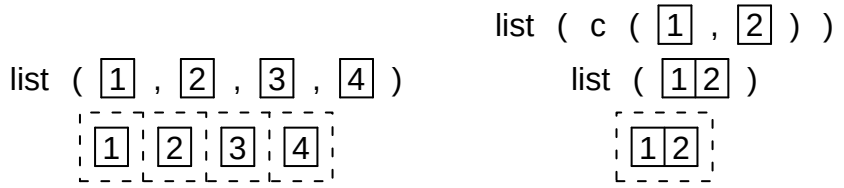


14 list

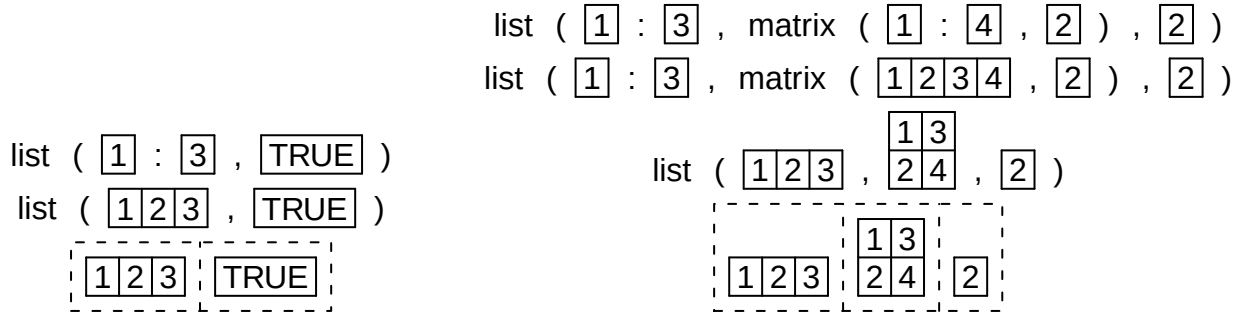
14.1 Creation, anatomy

Creating a list by enumerating its components.

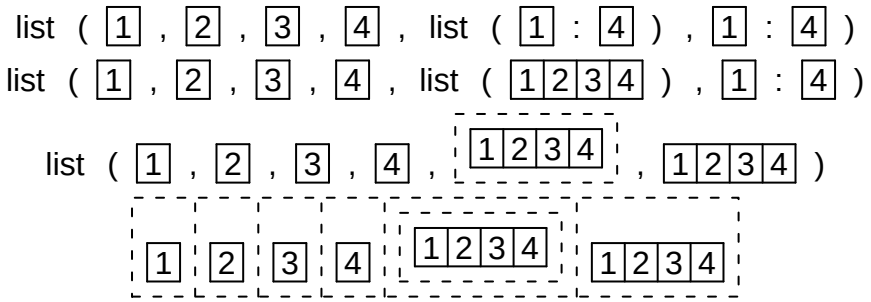
A list of length 4 : contains 4 vectors, each of length 1 (left) ; a list of length 1 : contains 1 vector of length 2 (right).



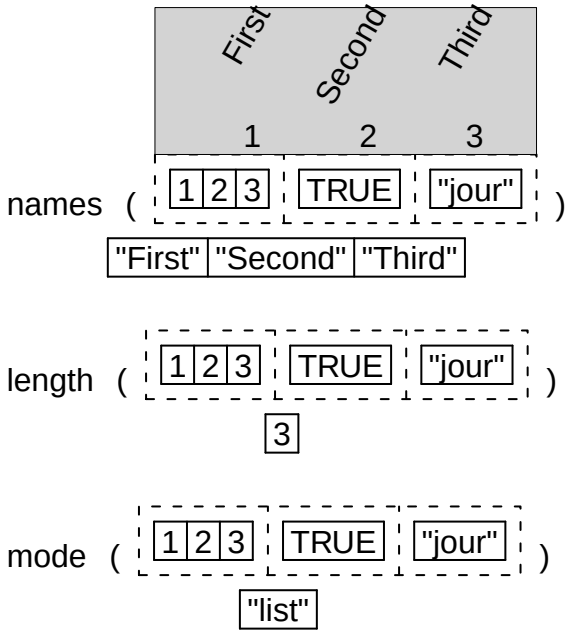
List can contain objects of different mode (left) ; it can contain objects of different dimensions (right).



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

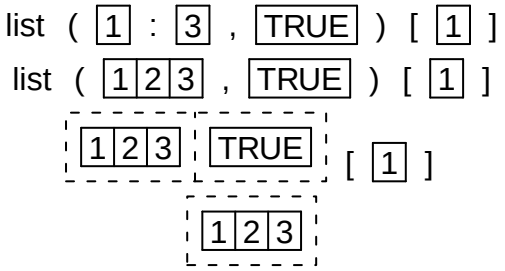


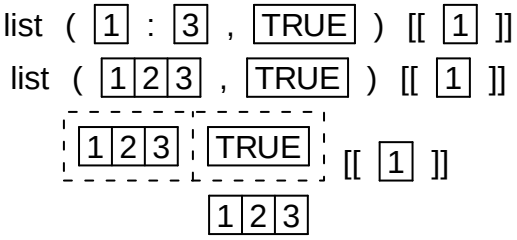
14.2 Basic functions



14.3 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.

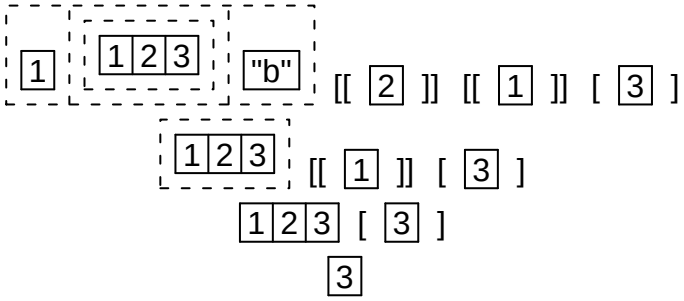




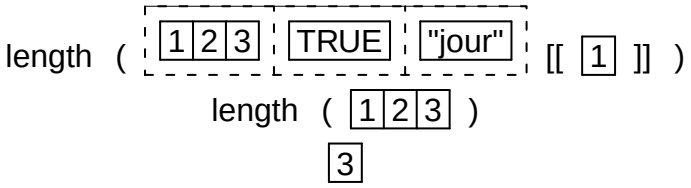
You can use vector of any length within the single-square bracket, while you can address only one element within the double-square-bracket operator, and then use only vector of length 1.

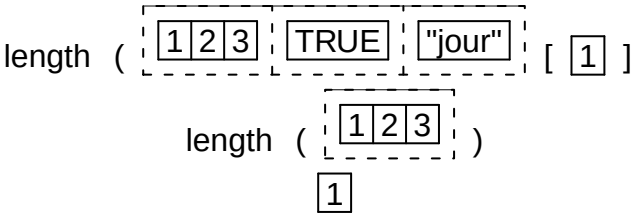
Since a list is a recursive data structure (may contain list), you can use several successive bracket operators in order to go down to the element you're interested in.

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

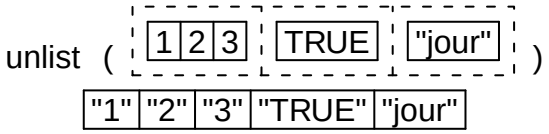


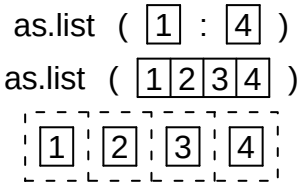
Be sure to understand the difference between "lengt(l[1])" and length(l[[1]])" :





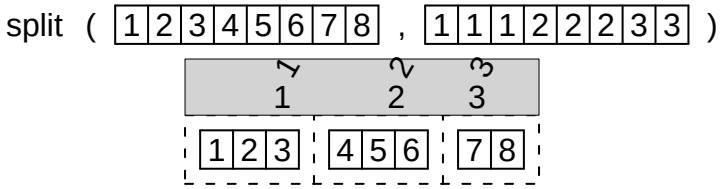
14.4 List and vector





14.5 List for expressing complex data structure

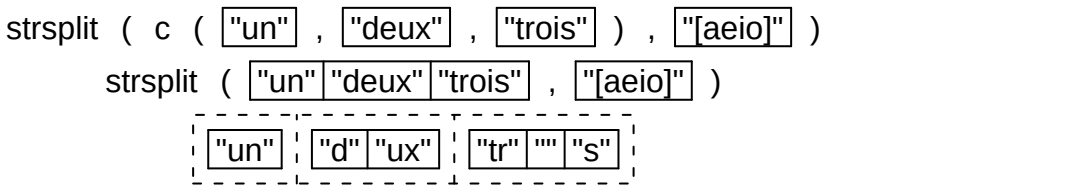
Grouping elements of a vector using a the level of a factors (see Factor) :

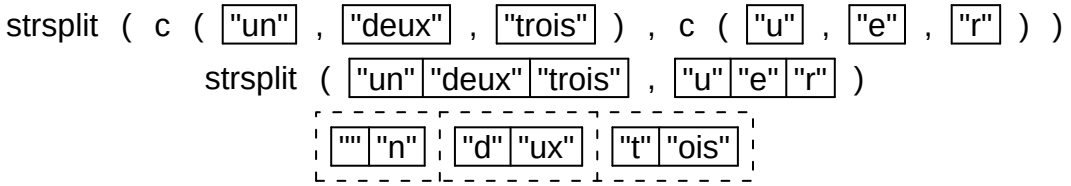


See also for instance strsplit() below.

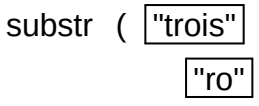
15 Regexp

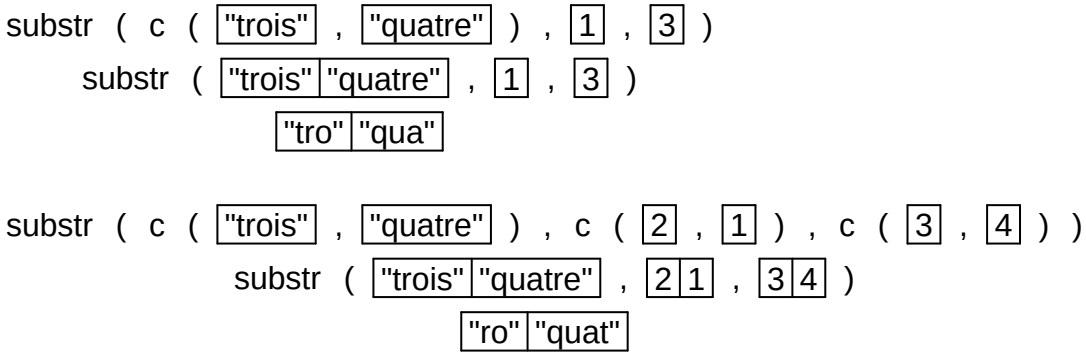
15.1 Split strings : strplit()



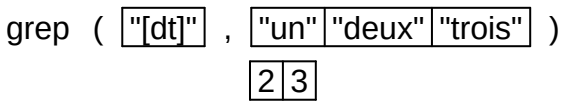


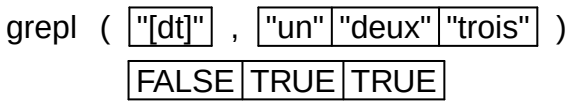
15.2 Extract sub strings : substr()



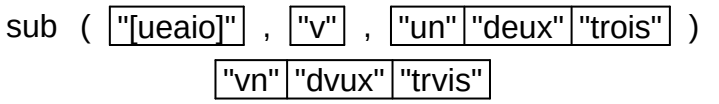


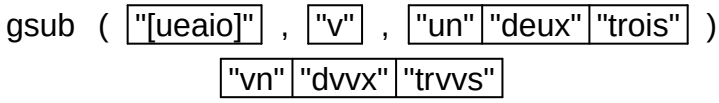
15.3 Searching elements of a character vector with regexp : grep()





15.4 Substitution : sub()



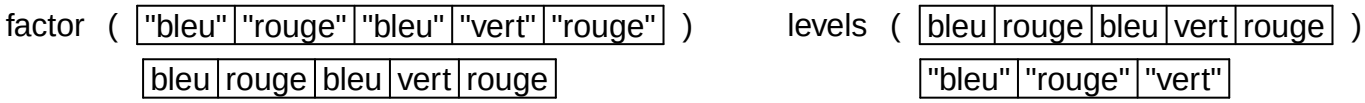


15.5 Searching substring in elements of character vector : regular expression

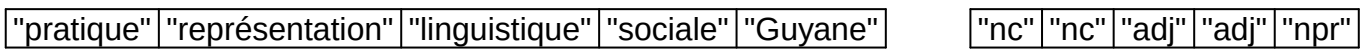
TODO

16 Factor

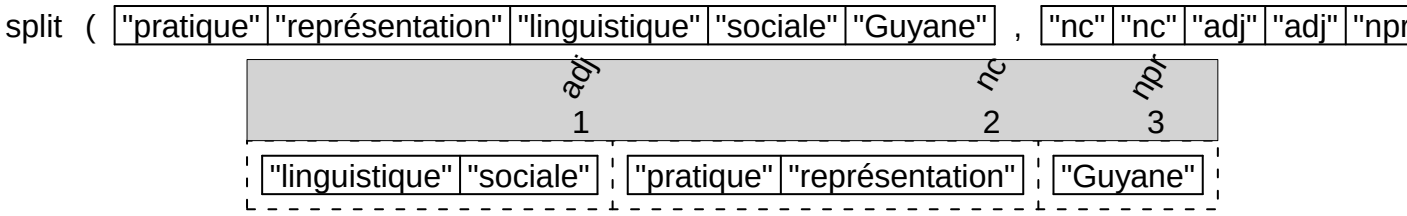
A factor represent a nominal random variables. It looks like a character vector, and may be created using a character vector (left). In this document, factors are represented without quotes around the values. The different values in the factor (the different modality in the random variable) may be retrieved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



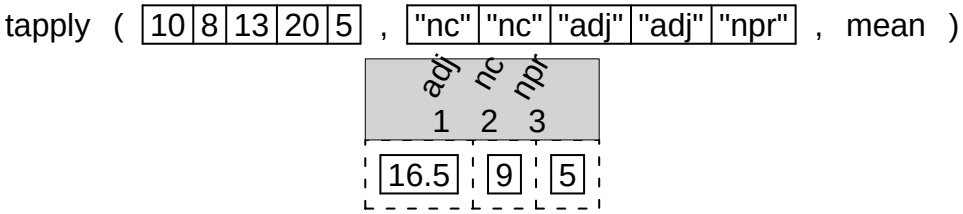
A factor is usefull for *grouping* elements. Suppose two vectors, one giving forms, and the other giving part of speech. You want to group the forms according to part of speech.



The function split() create a list, each element of the list corresponding to a group. The frist argument give the element to be grouped, the second one is a vector : it give the grouping.



You may give any type of vector 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 :



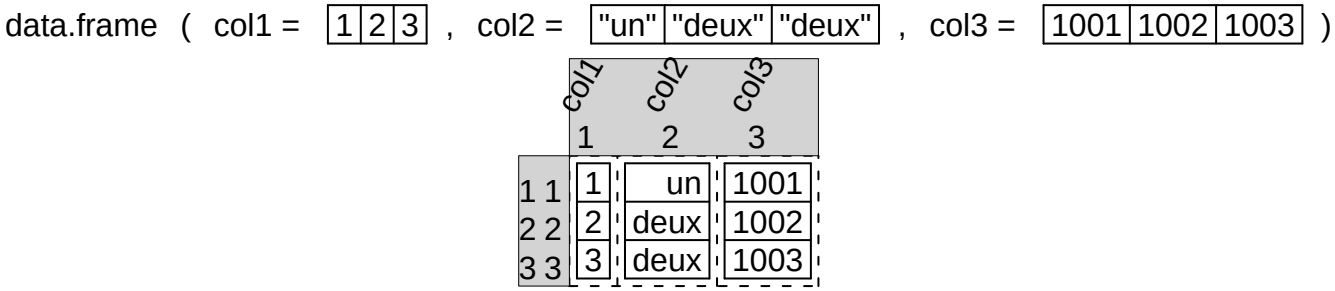
There is numerous functions using factor (or converting vector into factor), allowing for grouping (split(), rowsum(), tapply(), by(), etc.)

17 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 values, categorial values, boolean values, etc.

In a data frame, each row represent an individual, and each column represent a random variable. This is like a matrix, except that the column may have different type. This is like a list, since it may mix vectors of different type, but all vectors must have the same length. Data frame may often be seen as matrix, or as list.

From an internal representation, a data frame is a list of vector, each element of the list beeing a column. Thus, it is represented here with dotted line, like a list, and solid line in column :

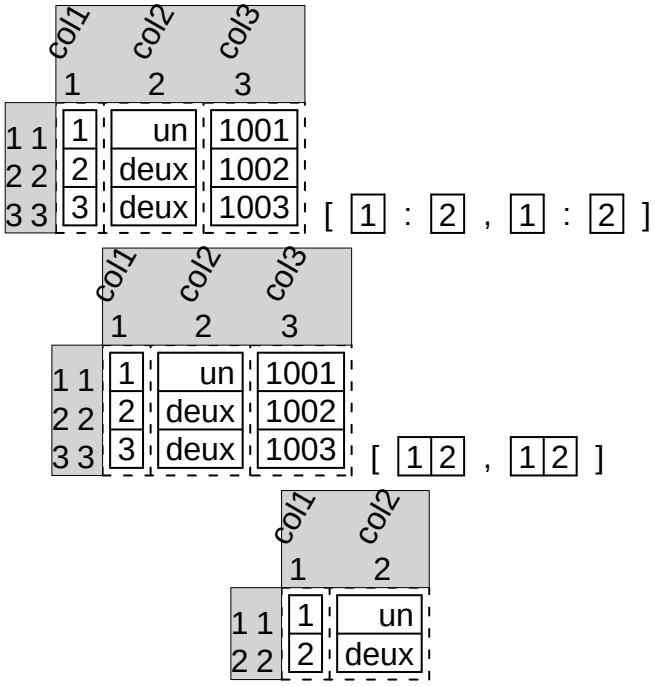


Data frame may be created by enumerating its columns.

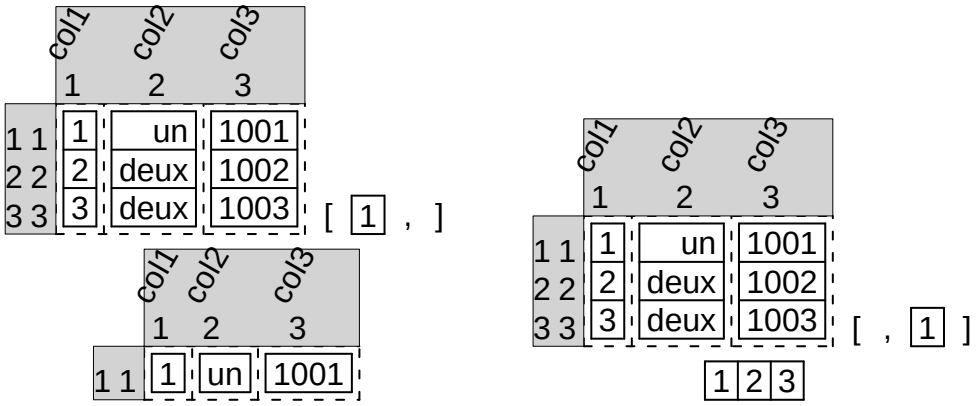
Like a matrix, a data frame may have rowname. In fact, unlike a matrix or a list, a data frame has *always* row and column name. Below, you see that automatic default row names have been added, corresponding to the index. rownames() and colnames() may be used with data frame as with matrix. Default colum name would have been added as well for unnamed column.

17.1 Extraction

Extracting is similar to extracting with matrix :



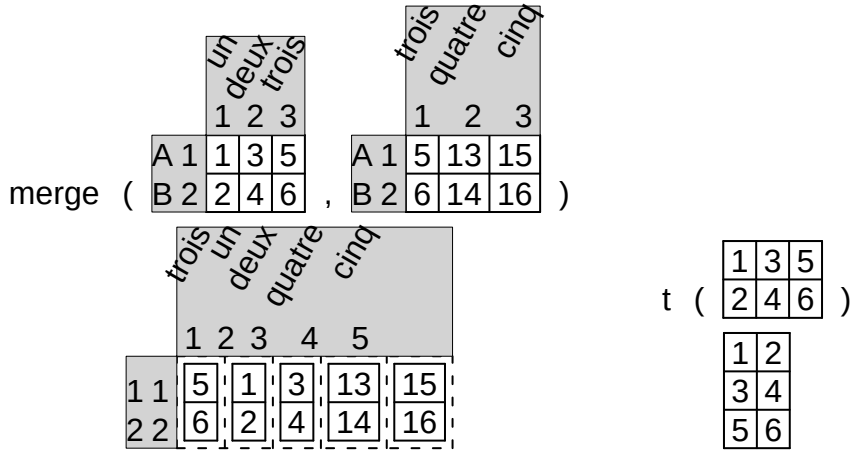
Note that row-extraction (left) and column-extraction (right) do not give the same data structure. Furthermore, in column extraction, the row name are lost :



17.2 Subseting

17.3 Merging

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



As shown by the graphical presentation, merge() may operate on both matrix or data frame, but create data frame. Columns to be used for combining may be given explicitly

TODO