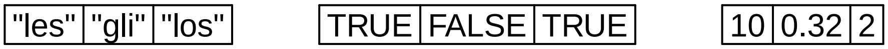
Visual R reference card (Draft)

23 janvier 2010

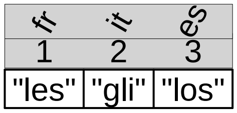
Sylvain Loiseau <sylvain.loiseau@unicaen.fr>
Université de Caen–Basse Normandie

1 Anatomy of a vector

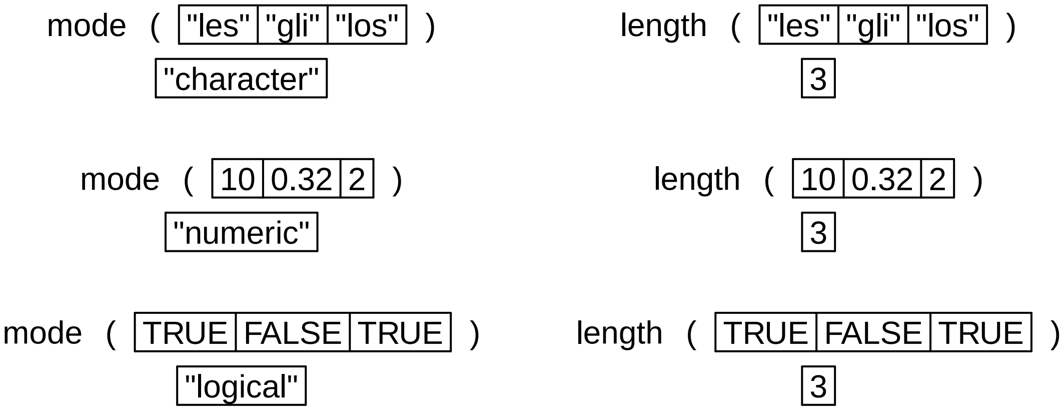
All elements of a vector have a common mode, one of logical, character, and numeric.



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



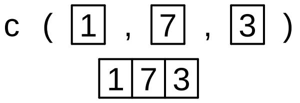
All vectors have two important properties : their mode and their length.

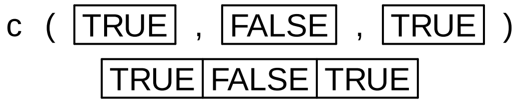


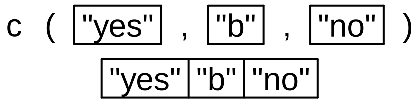
Functions are very precisly defined in terms of mode, number and length of vectors they may take as argument, and mode and length of vector they create.

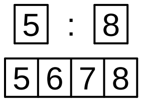
- length() take one vector of any mode and length and return a vector of numeric vector of length 1.
- mode() take one vector of any mode and length and return a character vector of length 1.

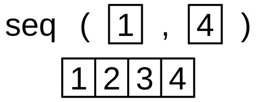
2 Creating a vector

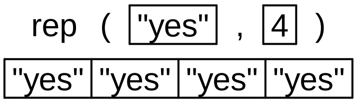


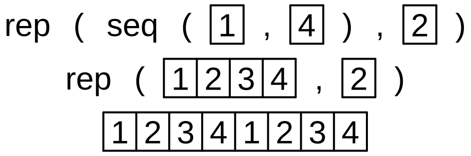








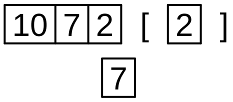


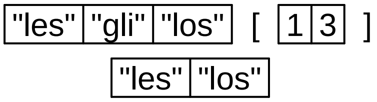


The function c() take any number of vectors of any length and any mode, but all vectors must have the same mode (see below, "Conversion"). It returns a vector of the same mode as the arguments and whose length is the sum of the length of its arguments.

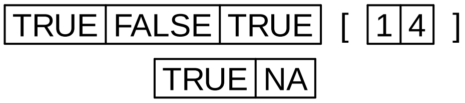
3 Extraction

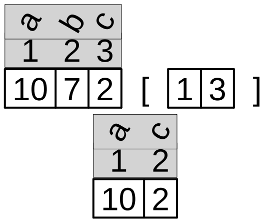
A vector can be created by extracting some elements of a vector.
Elements to be extracted can be addressed using their index.



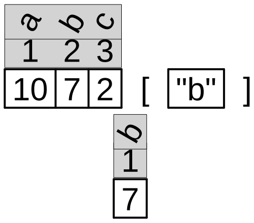


Index are 1-based. If an index is greater than the number of element in the vector, you get "NA". If the vector has names, their are preserved.





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))

```
|10|7|2| [ |TRUE|FALSE|TRUE| ]          |"a"|"b"|"c"|"d"| [ |TRUE|FALSE| ]
          |10|2|                                   |"a"|"c"|
```

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

```
|"a"|"b"|"c"|"d"| [ c ( |1| , |1| , |4| , |2| , |1| ) ]
          |"a"|"b"|"c"|"d"| [ |1|1|4|2|1| ]
                  |"a"|"a"|"d"|"b"|"a"|
```

4 Type conversion

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

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

5 Index

Some useful functions give index rather than the actual values.

```
which ( |TRUE|FALSE|FALSE|TRUE|FALSE|TRUE| )
                   |1|4|6|
```

```
                                        fr     it     es
                                        1      2      3
  order ( |10|2|7|15| )        order ( |"les"|"gli"|"los"| )
        |2|3|1|4|                        |2|1|3|

 which.min ( |2|1|3|4| )        which.max ( |2|1|3|4| )
              |2|                             |4|
```

6 Operator

Some numeric operators.

```
       |5| + |6|          |5| − |6|        |5| * |6|
         |11|               |−1|             |30|

       |5| / |6|          |5| < |6|        |5| >= |6|
  |0.833333333333333|      |TRUE|           |FALSE|
```

Some logical operators.

$$\boxed{5} == \boxed{6}$$
$$\boxed{\text{FALSE}}$$

$$\boxed{5} == \boxed{5}$$
$$\boxed{\text{TRUE}}$$

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

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

7 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.

$$\boxed{4}\boxed{3}\boxed{8} + \boxed{32}\boxed{3}\boxed{2}$$
$$\boxed{36}\boxed{6}\boxed{10}$$

$$\boxed{4}\boxed{3}\boxed{8} == \boxed{32}\boxed{3}\boxed{2}$$
$$\boxed{\text{FALSE}}\boxed{\text{TRUE}}\boxed{\text{FALSE}}$$

8 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 :

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

9 Some numerical functions

Some functions for numeric vectors.

$$\text{sum} \ (\ \text{c} \ (\boxed{2} , \boxed{1} , \boxed{3} , \boxed{4}) \ )$$
$$\text{sum} \ (\boxed{2}\boxed{1}\boxed{3}\boxed{4})$$
$$\boxed{10}$$

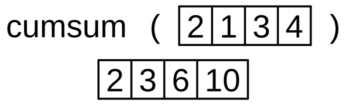
$$\text{mean} \ (\boxed{2}\boxed{1}\boxed{3}\boxed{4})$$
$$\boxed{2.5}$$

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

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

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

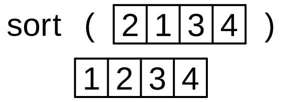
$$\text{min} \ (\boxed{2}\boxed{1}\boxed{3}\boxed{4})$$
$$\boxed{1}$$

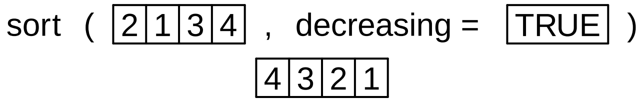


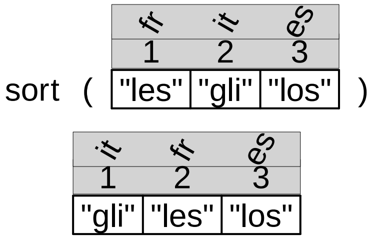
TODO : table

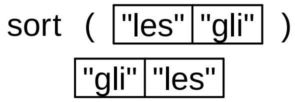
10 Sorting

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

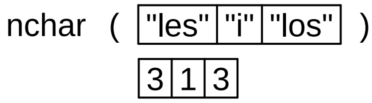




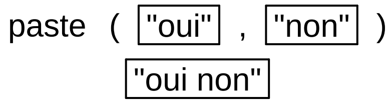


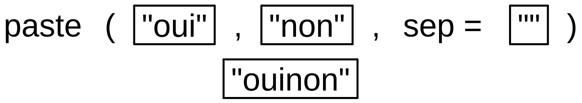


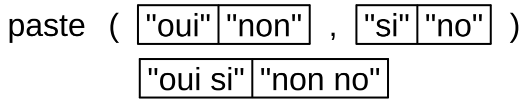
11 Some string functions

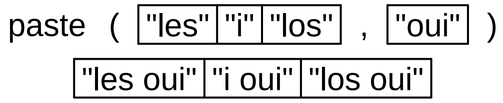


Recycling and vectorization are useful with paste(), which concatenates characters string at same index in several characters vectors :

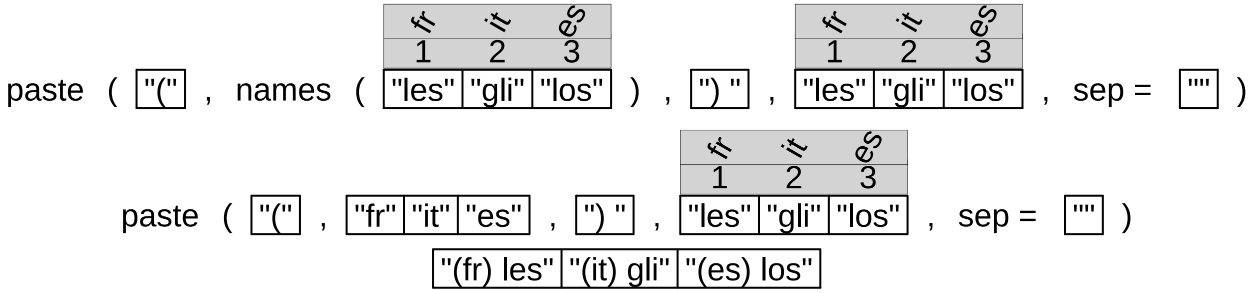







paste ("les" "i" "los" , "oui")
"les oui" "i oui" "los oui"

You can paste more than two vectors of characters :


fr it es
1 2 3
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"

12 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!

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

13 Factor

TODO

There is numerous situation where values of a vector are seen as modalities, allowing for grouping, etc. (split, rowsum, tapply, etc.)

14 Matrix

14.1 Creation

$$\text{matrix} \; (\; \boxed{1} : \boxed{6} \; , \; \text{nrow =} \; \boxed{3} \;)$$
$$\text{matrix} \; (\; \begin{array}{|c|c|c|c|c|c|}\hline 1&2&3&4&5&6\\\hline\end{array} \; , \; \text{nrow =} \; \boxed{3} \;)$$
$$\begin{array}{|c|c|}\hline 1&4\\\hline 2&5\\\hline 3&6\\\hline\end{array}$$

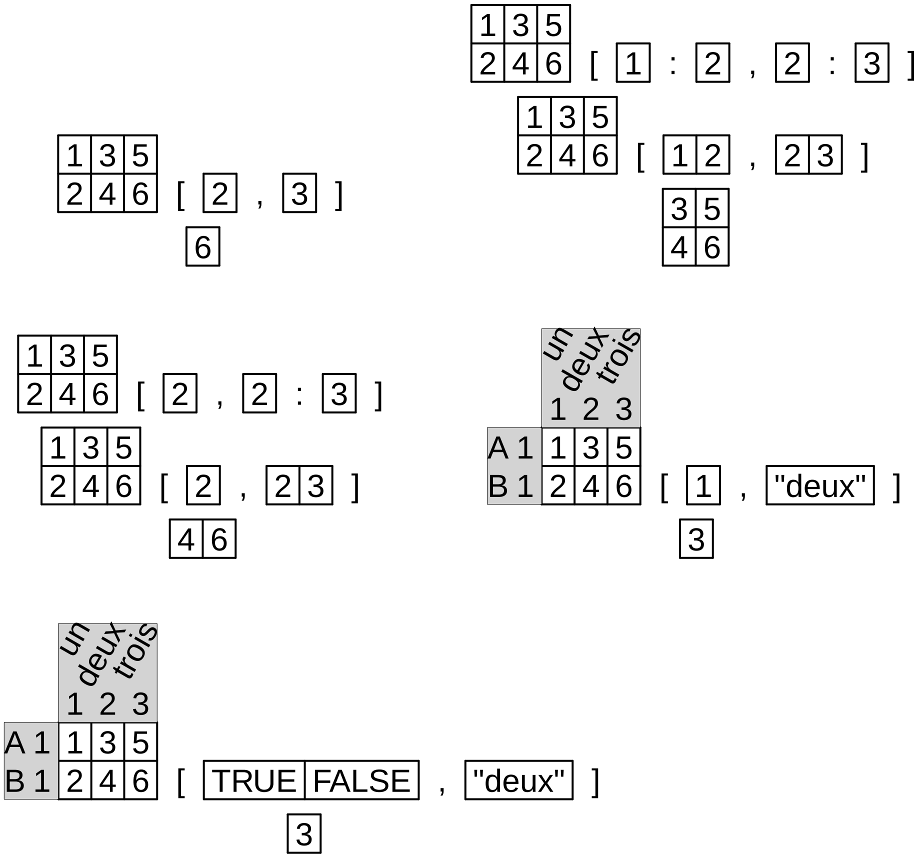
$$\text{matrix} \; (\; \boxed{1} : \boxed{6} \; , \; \boxed{3} \;)$$
$$\text{matrix} \; (\; \begin{array}{|c|c|c|c|c|c|}\hline 1&2&3&4&5&6\\\hline\end{array} \; , \; \boxed{3} \;)$$
$$\begin{array}{|c|c|}\hline 1&4\\\hline 2&5\\\hline 3&6\\\hline\end{array}$$

$$\text{matrix} \; (\; \boxed{1} : \boxed{6} \; , \; \boxed{3} \; , \; \text{byrow =} \; \boxed{\text{TRUE}} \;)$$
$$\text{matrix} \; (\; \begin{array}{|c|c|c|c|c|c|}\hline 1&2&3&4&5&6\\\hline\end{array} \; , \; \boxed{3} \; , \; \text{byrow =} \; \boxed{\text{TRUE}} \;)$$
$$\begin{array}{|c|c|}\hline 1&2\\\hline 3&4\\\hline 5&6\\\hline\end{array}$$

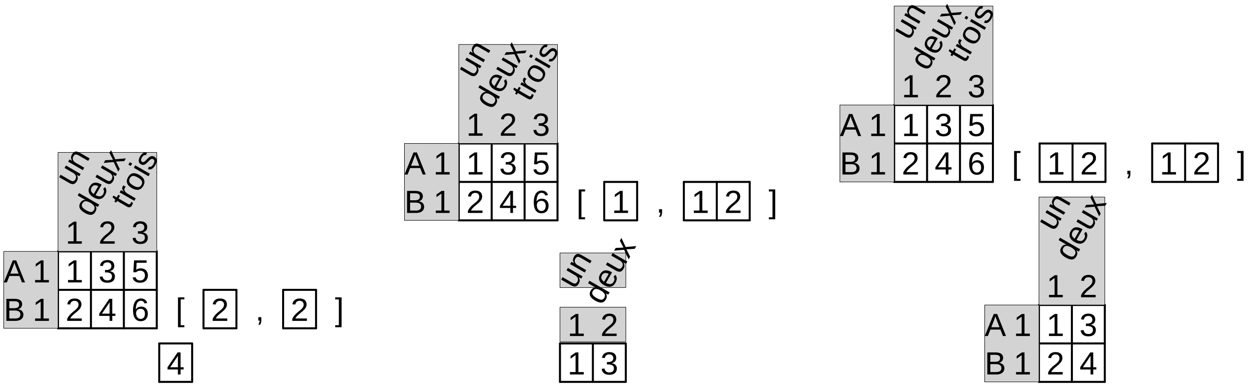
$$\text{matrix} \; (\; \boxed{1} : \boxed{6} \; , \; \text{ncol =} \; \boxed{3} \;)$$
$$\text{matrix} \; (\; \begin{array}{|c|c|c|c|c|c|}\hline 1&2&3&4&5&6\\\hline\end{array} \; , \; \text{ncol =} \; \boxed{3} \;)$$
$$\begin{array}{|c|c|c|}\hline 1&3&5\\\hline 2&4&6\\\hline\end{array}$$

$$\text{matrix} \; (\; \boxed{\text{TRUE}} \; , \; \text{nrow =} \; \boxed{2} \; , \; \text{ncol =} \; \boxed{3} \;)$$
$$\begin{array}{|c|c|c|}\hline \text{TRUE}&\text{TRUE}&\text{TRUE}\\\hline \text{TRUE}&\text{TRUE}&\text{TRUE}\\\hline\end{array}$$

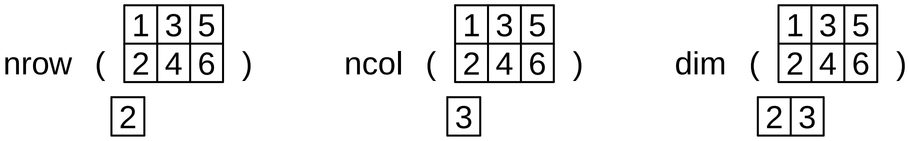
14.2 Extraction with a matrix



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



$$\text{rownames}\ \left(\begin{array}{c|c|c|c|}
 & \text{un} & \text{deux} & \text{trois}\\
 & 1 & 2 & 3\\ \hline
\text{A 1} & 1 & 3 & 5\\ \hline
\text{B 1} & 2 & 4 & 6\\ \hline
\end{array}\right)
\qquad
\text{colnames}\ \left(\begin{array}{c|c|c|c|}
 & \text{un} & \text{deux} & \text{trois}\\
 & 1 & 2 & 3\\ \hline
\text{A 1} & 1 & 3 & 5\\ \hline
\text{B 1} & 2 & 4 & 6\\ \hline
\end{array}\right)$$

$$\begin{array}{|c|c|}\hline \text{"A"} & \text{"B"}\\ \hline\end{array}
\qquad
\begin{array}{|c|c|c|}\hline \text{"un"} & \text{"deux"} & \text{"trois"}\\ \hline\end{array}$$

14.4 Summing a matrix

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

$$\boxed{21}$$

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

$$\begin{array}{|c|c|}\hline 9 & 12\\ \hline\end{array}
\qquad
\begin{array}{|c|c|c|}\hline 3 & 7 & 11\\ \hline\end{array}$$

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

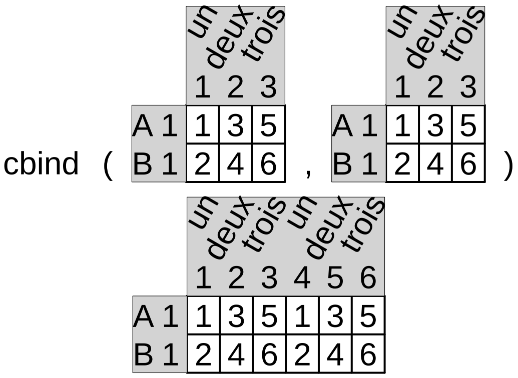
$$\text{rowsum}\ \left(\begin{array}{|c|r|}\hline 1&6\\ \hline 2&7\\ \hline 3&8\\ \hline 4&9\\ \hline 5&10\\ \hline\end{array}\ ,\ \begin{array}{|c|c|c|c|c|}\hline 2&1&2&1&3\\ \hline\end{array}\right)$$

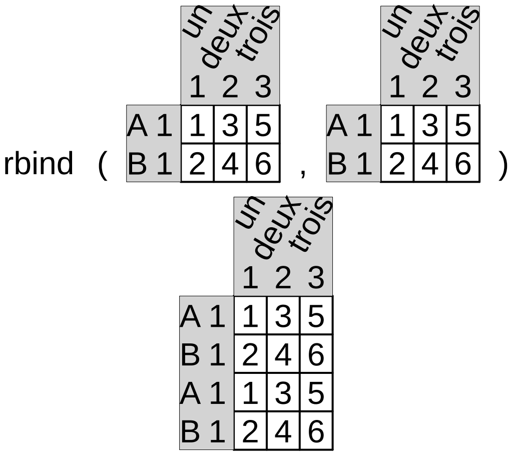
$$\begin{array}{|c|c|}\hline 6&16\\ \hline 4&14\\ \hline 5&10\\ \hline\end{array}$$

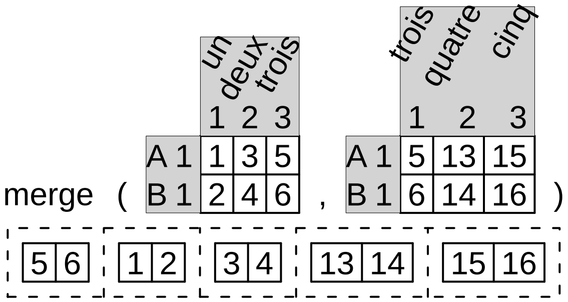
14.5 Changing

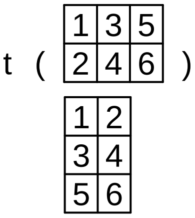
$$\text{as.vector}\ \left(\begin{array}{|c|c|c|}\hline 1&3&5\\ \hline 2&4&6\\ \hline\end{array}\right)
\qquad
\begin{array}{|c|c|c|}\hline 1&3&5\\ \hline 2&4&6\\ \hline\end{array}\ +\ \boxed{3}$$

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


cbind (A 1 B 1 un deux trois 1 2 3 1 3 5 2 4 6 , A 1 B 1 un deux trois 1 2 3 1 3 5 2 4 6)
un deux trois un deux trois
1 2 3 4 5 6
A 1 1 3 5 1 3 5
B 1 2 4 6 2 4 6


rbind (A 1 B 1 un deux trois 1 2 3 1 3 5 2 4 6 , A 1 B 1 un deux trois 1 2 3 1 3 5 2 4 6)
un deux trois
1 2 3
A 1 1 3 5
B 1 2 4 6
A 1 1 3 5
B 1 2 4 6


merge (A 1 B 1 un deux trois 1 2 3 1 3 5 2 4 6 , A 1 B 1 trois quatre cinq 1 2 3 5 13 15 6 14 16)
5 6
1 2
3 4
13 14
15 16

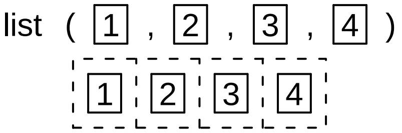


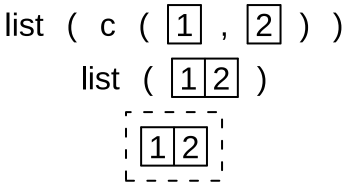
15 list

15.1 Creation, anatomy

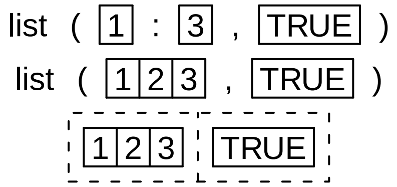
Creating a list by enumerating its components.

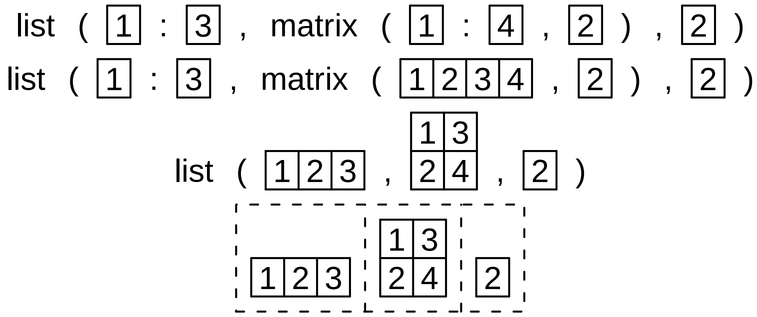
A list of length 4 : contains 4 vectors, each of length 1 (left) ; a list of length 1 : contains 1 vector of length 2 (right).



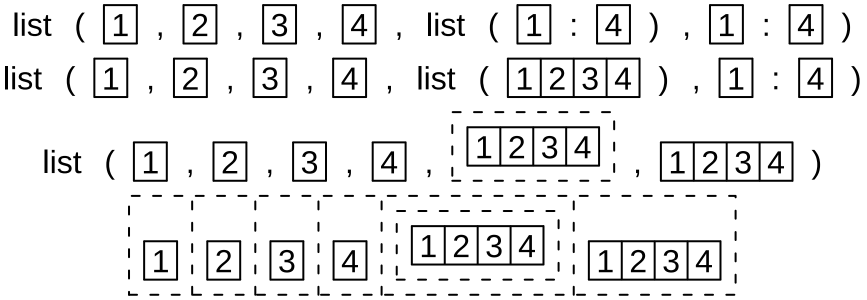


List can contain objects of different mode (left) ; it can contain objects of different dimensions (right).



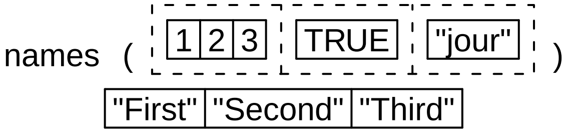

list (1 : 3 , matrix (1 : 4 , 2) , 2)
list (1 : 3 , matrix (1 2 3 4 , 2) , 2)
list (1 2 3 , 1 3 2 4 , 2)
1 2 3 1 3 2 4 2

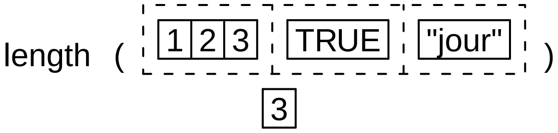
A list can be recursive : a component may be a list.

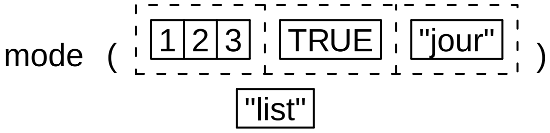

list (1 , 2 , 3 , 4 , list (1 : 4) , 1 : 4)
list (1 , 2 , 3 , 4 , list (1 2 3 4) , 1 : 4)
list (1 , 2 , 3 , 4 , 1 2 3 4 , 1 2 3 4)
1 2 3 4 1 2 3 4 1 2 3 4

15.2 Basic functions

TODO...

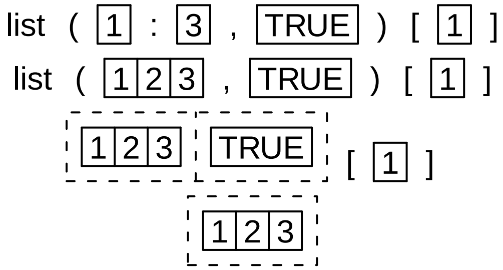


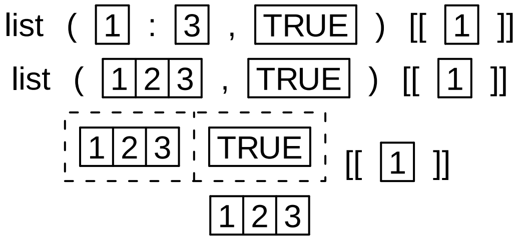




15.3 Extraction

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 extract elements from a vector), while the second is completely different : it give the content of one (and only one) element of a list.

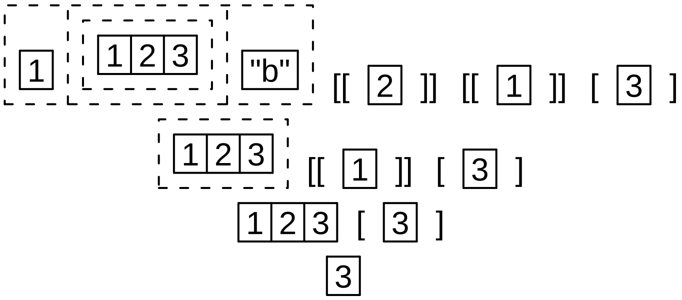




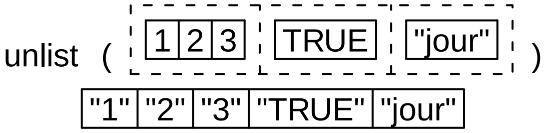
You can use vector of any length within the single-square bracket, while you can address only one element within the double-square-bracket operator, and then use only vector of length 1.

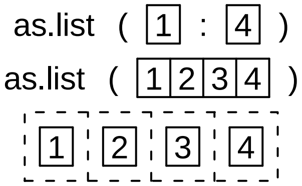
Since a list is a recursive data structure (may contain list), you can use several successive bracket operators in order to go down to the element you're interested in.

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



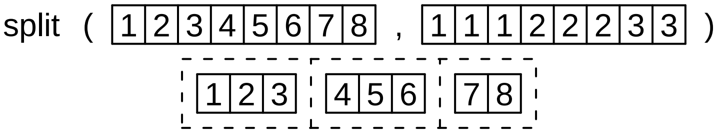
15.4 List and vector





15.5 List for expressing complex data structure

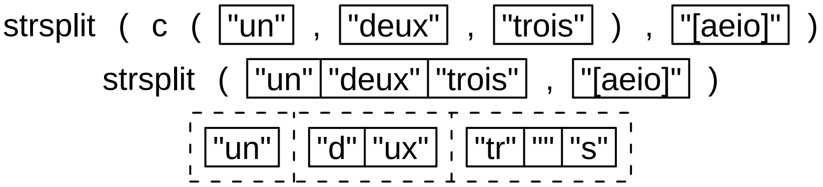
Grouping elements of a vector using a the level of a factors :



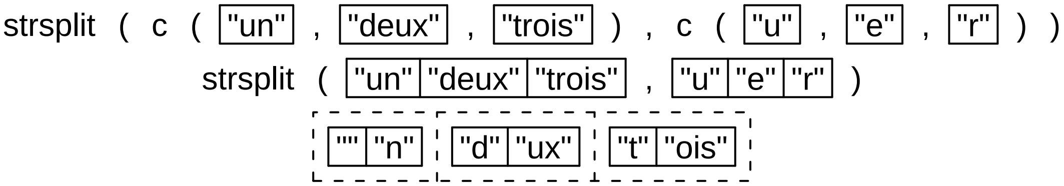
See also for instance strsplit() below.

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()

$$\text{substr} \; (\; \boxed{\text{"trois"}} \; , \; \boxed{2} \; , \; \boxed{3} \;)$$
$$\boxed{\text{"ro"}}$$

$$\text{substr} \; (\; \text{c} \; (\; \boxed{\text{"trois"}} \; , \; \boxed{\text{"quatre"}} \;) \; , \; \boxed{1} \; , \; \boxed{3} \;)$$
$$\text{substr} \; (\; \boxed{\text{"trois"}}\boxed{\text{"quatre"}} \; , \; \boxed{1} \; , \; \boxed{3} \;)$$
$$\boxed{\text{"tro"}}\boxed{\text{"qua"}}$$

$$\text{substr} \; (\; \text{c} \; (\; \boxed{\text{"trois"}} \; , \; \boxed{\text{"quatre"}} \;) \; , \; \text{c} \; (\; \boxed{2} \; , \; \boxed{1} \;) \; , \; \text{c} \; (\; \boxed{3} \; , \; \boxed{4} \;) \;)$$
$$\text{substr} \; (\; \boxed{\text{"trois"}}\boxed{\text{"quatre"}} \; , \; \boxed{2}\boxed{1} \; , \; \boxed{3}\boxed{4} \;)$$
$$\boxed{\text{"ro"}}\boxed{\text{"quat"}}$$

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

$$\text{grep} \; (\; \boxed{\text{"[dt]"}} \; , \; \boxed{\text{"un"}}\boxed{\text{"deux"}}\boxed{\text{"trois"}} \;)$$
$$\boxed{2}\boxed{3}$$

$$\text{grepl} \; (\; \boxed{\text{"[dt]"}} \; , \; \boxed{\text{"un"}}\boxed{\text{"deux"}}\boxed{\text{"trois"}} \;)$$
$$\boxed{\text{FALSE}}\boxed{\text{TRUE}}\boxed{\text{TRUE}}$$

16.4 Substitution : sub()

$$\text{sub} \; (\; \boxed{\text{"[ueaio]"}} \; , \; \boxed{\text{"v"}} \; , \; \boxed{\text{"un"}}\boxed{\text{"deux"}}\boxed{\text{"trois"}} \;)$$
$$\boxed{\text{"vn"}}\boxed{\text{"dvux"}}\boxed{\text{"trvis"}}$$

$$\text{gsub} \; (\; \boxed{\text{"[ueaio]"}} \; , \; \boxed{\text{"v"}} \; , \; \boxed{\text{"un"}}\boxed{\text{"deux"}}\boxed{\text{"trois"}} \;)$$
$$\boxed{\text{"vn"}}\boxed{\text{"dvvx"}}\boxed{\text{"trvvs"}}$$

16.5 Searching substring in elements of character vector : regexpr()

TODO

17 Data Frame

TODO