|  |  |
| --- | --- |
| cbind {base} | R Documentation |

**Combine R Objects by Rows or Columns**

**Description**

Take a sequence of vector, matrix or data-frame arguments and combine by *c*olumns or *r*ows, respectively. These are generic functions with methods for other **R** classes.

**Usage**

cbind(..., deparse.level = 1)

rbind(..., deparse.level = 1)

## S3 method for class 'data.frame'

rbind(..., deparse.level = 1, make.row.names = TRUE,

stringsAsFactors = default.stringsAsFactors())

**Arguments**

|  |  |
| --- | --- |
| ... | (generalized) vectors or matrices. These can be given as named arguments. Other **R** objects may be coerced as appropriate, or S4 methods may be used: see sections ‘Details’ and ‘Value’. (For the "data.frame" method of cbind these can be further arguments to [data.frame](http://127.0.0.1:14695/library/base/help/data.frame) such as stringsAsFactors.) |
| deparse.level | integer controlling the construction of labels in the case of non-matrix-like arguments (for the default method):  deparse.level = 0 constructs no labels; the default,  deparse.level = 1 or 2 constructs labels from the argument names, see the ‘Value’ section below. |
| make.row.names | (only for data frame method:) logical indicating if unique and valid [row.names](http://127.0.0.1:14695/library/base/help/row.names) should be constructed from the arguments. |
| stringsAsFactors | logical, passed to [as.data.frame](http://127.0.0.1:14695/library/base/help/as.data.frame); only has an effect when the ... arguments contain a (non-data.frame) [character](http://127.0.0.1:14695/library/base/help/character). |

**Details**

The functions cbind and rbind are S3 generic, with methods for data frames. The data frame method will be used if at least one argument is a data frame and the rest are vectors or matrices. There can be other methods; in particular, there is one for time series objects. See the section on ‘Dispatch’ for how the method to be used is selected. If some of the arguments are of an S4 class, i.e., [isS4](http://127.0.0.1:14695/library/base/help/isS4)(.) is true, S4 methods are sought also, and the hidden cbind / rbind functions from package **methods** maybe called, which in turn build on [cbind2](http://127.0.0.1:14695/library/base/help/cbind2) or [rbind2](http://127.0.0.1:14695/library/base/help/rbind2), respectively. In that case, deparse.level is obeyed, similarly to the default method.

In the default method, all the vectors/matrices must be atomic (see [vector](http://127.0.0.1:14695/library/base/help/vector)) or lists. Expressions are not allowed. Language objects (such as formulae and calls) and pairlists will be coerced to lists: other objects (such as names and external pointers) will be included as elements in a list result. Any classes the inputs might have are discarded (in particular, factors are replaced by their internal codes).

If there are several matrix arguments, they must all have the same number of columns (or rows) and this will be the number of columns (or rows) of the result. If all the arguments are vectors, the number of columns (rows) in the result is equal to the length of the longest vector. Values in shorter arguments are recycled to achieve this length (with a [warning](http://127.0.0.1:14695/library/base/help/warning) if they are recycled only *fractionally*).

When the arguments consist of a mix of matrices and vectors the number of columns (rows) of the result is determined by the number of columns (rows) of the matrix arguments. Any vectors have their values recycled or subsetted to achieve this length.

For cbind (rbind), vectors of zero length (including NULL) are ignored unless the result would have zero rows (columns), for S compatibility. (Zero-extent matrices do not occur in S3 and are not ignored in **R**.)

Matrices are restricted to less than *2^31* rows and columns even on 64-bit systems. So input vectors have the same length restriction: as from **R** 3.2.0 input matrices with more elements (but meeting the row and column restrictions) are allowed.

**Value**

For the default method, a matrix combining the ... arguments column-wise or row-wise. (Exception: if there are no inputs or all the inputs are NULL, the value is NULL.)

The type of a matrix result determined from the highest type of any of the inputs in the hierarchy raw < logical < integer < double < complex < character < list .

For cbind (rbind) the column (row) names are taken from the colnames (rownames) of the arguments if these are matrix-like. Otherwise from the names of the arguments or where those are not supplied and deparse.level > 0, by deparsing the expressions given, for deparse.level = 1 only if that gives a sensible name (a ‘symbol’, see [is.symbol](http://127.0.0.1:14695/library/base/help/is.symbol)).

For cbind row names are taken from the first argument with appropriate names: rownames for a matrix, or names for a vector of length the number of rows of the result.

For rbind column names are taken from the first argument with appropriate names: colnames for a matrix, or names for a vector of length the number of columns of the result.

**Data frame methods**

The cbind data frame method is just a wrapper for [data.frame](http://127.0.0.1:14695/library/base/help/data.frame)(..., check.names = FALSE). This means that it will split matrix columns in data frame arguments, and convert character columns to factors unless stringsAsFactors = FALSE is specified.

The rbind data frame method first drops all zero-column and zero-row arguments. (If that leaves none, it returns the first argument with columns otherwise a zero-column zero-row data frame.) It then takes the classes of the columns from the first data frame, and matches columns by name (rather than by position). Factors have their levels expanded as necessary (in the order of the levels of the levelsets of the factors encountered) and the result is an ordered factor if and only if all the components were ordered factors. (The last point differs from S-PLUS.) Old-style categories (integer vectors with levels) are promoted to factors.

**Dispatch**

The method dispatching is *not* done via [UseMethod](http://127.0.0.1:14695/library/base/help/UseMethod)(), but by C-internal dispatching. Therefore there is no need for, e.g., rbind.default.

The dispatch algorithm is described in the source file (‘.../src/main/bind.c’) as

1. For each argument we get the list of possible class memberships from the class attribute.
2. We inspect each class in turn to see if there is an applicable method.
3. If we find an applicable method we make sure that it is identical to any method determined for prior arguments. If it is identical, we proceed, otherwise we immediately drop through to the default code.

If you want to combine other objects with data frames, it may be necessary to coerce them to data frames first. (Note that this algorithm can result in calling the data frame method if all the arguments are either data frames or vectors, and this will result in the coercion of character vectors to factors.)

**References**

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

**See Also**

[c](http://127.0.0.1:14695/library/base/help/c) to combine vectors (and lists) as vectors, [data.frame](http://127.0.0.1:14695/library/base/help/data.frame) to combine vectors and matrices as a data frame.

**Examples**

m <- cbind(1, 1:7) # the '1' (= shorter vector) is recycled

m

m <- cbind(m, 8:14)[, c(1, 3, 2)] # insert a column

m

cbind(1:7, diag(3)) # vector is subset -> warning

cbind(0, rbind(1, 1:3))

cbind(I = 0, X = rbind(a = 1, b = 1:3)) # use some names

xx <- data.frame(I = rep(0,2))

cbind(xx, X = rbind(a = 1, b = 1:3)) # named differently

cbind(0, matrix(1, nrow = 0, ncol = 4)) #> Warning (making sense)

dim(cbind(0, matrix(1, nrow = 2, ncol = 0))) #-> 2 x 1

## deparse.level

dd <- 10

rbind(1:4, c = 2, "a++" = 10, dd, deparse.level = 0) # middle 2 rownames

rbind(1:4, c = 2, "a++" = 10, dd, deparse.level = 1) # 3 rownames (default)

rbind(1:4, c = 2, "a++" = 10, dd, deparse.level = 2) # 4 rownames

## cheap row names:

b0 <- gl(3,4, labels=letters[1:3])

bf <- setNames(b0, paste0("o", seq\_along(b0)))

df <- data.frame(a = 1, B = b0, f = gl(4,3))

df. <- data.frame(a = 1, B = bf, f = gl(4,3))

new <- data.frame(a = 8, B ="B", f = "1")

(df1 <- rbind(df , new))

(df.1 <- rbind(df., new))

stopifnot(identical(df1, rbind(df, new, make.row.names=FALSE)),

identical(df1, rbind(df., new, make.row.names=FALSE)))