

Package ‘abind’

April 17, 2009

Version 1.1-0

Date 2004-03-12

Title Combine multi-dimensional arrays

Author Tony Plate <tplate@acm.org> and Richard Heiberger

Maintainer Tony Plate <tplate@acm.org>

Description Combine multi-dimensional arrays. This is a generalization of cbind and rbind. Takes a sequence of vectors, matrices, or arrays and produces a single array of the same or higher dimension.

Depends R (>= 1.5.0)

License LGPL (>= 2.0)

Repository CRAN

Date/Publication 2004-03-15 15:54:08

R topics documented:

abind	1
adrop	4
Index	6

 abind

Combine multi-dimensional arrays

Description

Combine multi-dimensional arrays. This is a generalization of `cbind` and `rbind`. Takes a sequence of vectors, matrices, or arrays and produces a single array of the same or higher dimension.

Usage

```
abind(..., along=N, rev.along=NULL, new.names=NULL, force.array=TRUE, make.names=us
```

Arguments

- `...` Any number of vectors, matrices, arrays, or data frames. The dimensions of all the arrays must match, except on one dimension (specified by `along=`). If these arguments are named, the name will be used for the name of the dimension along which the arrays are joined. Vectors are treated as having a `dim` attribute of length one.
- Alternatively, there can be one (and only one) list argument supplied, whose components are the objects to be bound together. Names of the list components are treated in the same way as argument names.
- `along` (optional) The dimension along which to bind the arrays. The default is the last dimension, i.e., the maximum length of the `dim` attribute of the supplied arrays. `along=` can take any non-negative value up to the minimum length of the `dim` attribute of supplied arrays plus one. When `along=` has a fractional value, a value less than 1, or a value greater than `N` (`N` is the maximum of the lengths of the `dim` attribute of the objects to be bound together), a new dimension is created in the result. In these cases, the dimensions of all arguments must be identical.
- `rev.along` (optional) Alternate way to specify the dimension along which to bind the arrays: `along = N + 1 - rev.along`. This is provided mainly to allow easy specification of `along = N + 1` (by supplying `rev.along=0`). If both `along` and `rev.along` are supplied, the supplied value of `along` is ignored.
- `new.names` (optional) If `new.names` is a list, it is the first choice for the `dimnames` attribute of the result. It should have the same structure as a `dimnames` attribute. If the names for a particular dimension are `NULL`, names for this dimension are constructed in other ways.
- If `new.names` is a character vector, it is used for dimension names in the same way as argument names are used. Zero length (`""`) names are ignored.
- `force.array` (optional) If `FALSE`, `rbind` or `cbind` are called when possible, i.e., when the arguments are all vectors, and `along` is not 1, or when the arguments are vectors or matrices or data frames and `along` is 1 or 2. If `rbind` or `cbind` are used, they will preserve the `data.frame` classes (or any other class that `r/cbind` preserve). Otherwise, `abind` will convert objects to class `array`. Thus, to guarantee that

an array object is returned, supply the argument `force.array=TRUE`. Note that the use of `rbind` or `cbind` introduces some subtle changes in the way default dimension names are constructed: see the examples below.

`make.names` (optional) If `TRUE`, the last resort for `dimnames` for the `along` dimension will be the deparsed versions of anonymous arguments. This can result in cumbersome names when arguments are expressions.
<p>The default is `FALSE`.

`use.anon.names` (optional) `use.anon.names` is a deprecated synonym for `make.names`.

`use.first.dimnames` (optional) When dimension names are present on more than one argument, should dimension names for the result be take from the first available (the default is to take them from the last available, which is the same behavior as `rbind` and `cbind`.)

Details

The dimensions of the supplied vectors or arrays do not need to be identical, e.g., arguments can be a mixture of vectors and matrices. `abind` coerces arguments by the addition of one dimension in order to make them consistent with other arguments and `along=`. The extra dimension is added in the place specified by `along=`.

The default action of `abind` is to concatenate on the last dimension, rather than increase the number of dimensions. For example, the result of calling `abind` with vectors is a longer vector (see first example below). This differs from the action of `rbind` and `cbind` which is to return a matrix when called with vectors. `abind` can be made to behave like `cbind` on vectors by specifying `along=2`, and like `rbind` by specifying `along=0`.

The `dimnames` of the returned object are pieced together from the `dimnames` of the arguments, and the names of the arguments. Names for each dimension are searched for in the following order: `new.names`, argument name, `dimnames` (or names) attribute of last argument, `dimnames` (or names) attribute of second last argument, etc. (Supplying the argument `use.first.dimnames=TRUE` changes this to cause `abind` to use `dimnames` or names from the first argument first. The default behavior is the same as for `rbind` and `cbind`: use `dimnames` from later arguments.) If some names are supplied for the `along` dimension (either as argument names or `dimnames` in arguments), names are constructed for anonymous arguments unless `use.anon.names=FALSE`.

Value

An array with a `dim` attribute calculated as follows.

Let $rMin = \min(\text{sapply}(\text{list}(\dots), \text{function}(x) \text{ length}(\text{dim}(x))))$ and $rMax = \max(\text{sapply}(\text{list}(\dots), \text{function}(x) \text{ length}(\text{dim}(x))))$ (where the length of the dimensions of a vector are taken to be 1). Then $rMax$ should be equal to or one greater than $rMin$.

If `along` refers to an existing dimension, then the length of the `dim` attribute of the result is $rMax$. If `along` does not refer to an existing dimension, then $rMax$ should equal $rMin$ and the length of the `dim` attribute of the result will be $rMax+1$.

`rbind` or `cbind` are called to compute the result if (a) `force.array=FALSE`; and (b) the result will be a two-dimensional object.

Author(s)

Tony Plate (tplate@acm.org) and Richard Heiberger

Examples

```
# Five different ways of binding together two matrices
x <- matrix(1:12,3,4)
y <- x+100
dim(abind(x,y,along=0))      # binds on new dimension before first
dim(abind(x,y,along=1))      # binds on first dimension
dim(abind(x,y,along=1.5))
dim(abind(x,y,along=2))
dim(abind(x,y,along=3))
dim(abind(x,y,rev.along=1)) # binds on last dimension
dim(abind(x,y,rev.along=0)) # binds on new dimension after last

# Unlike cbind or rbind in that the default is to bind
# along the last dimension of the inputs, which for vectors
# means the result is a vector (because a vector is
# treated as an array with length(dim(x))==1).
abind(x=1:4,y=5:8)
# Like cbind
abind(x=1:4,y=5:8,along=2)
abind(x=1:4,matrix(5:20,nrow=4),along=2)
abind(1:4,matrix(5:20,nrow=4),along=2)
# Like rbind
abind(x=1:4,matrix(5:20,nrow=4),along=1)
abind(1:4,matrix(5:20,nrow=4),along=1)
# Create a 3-d array out of two matrices
abind(x=matrix(1:16,nrow=4),y=matrix(17:32,nrow=4),along=3)
```

adrop

Drop dimensions of an array object

Description

Drop degenerate dimensions of an array object. Offers more control than the `drop()` function.

Usage

```
adrop(x, drop = TRUE, named.vector = TRUE, one.d.array = FALSE)
```

Arguments

<code>x</code>	An array (including a matrix)
<code>drop</code>	A logical or numeric vector describing the dimensions to be dropped.
<code>named.vector</code>	(Optional, defaults to <code>TRUE</code> . Controls whether a vector result has names derived from the <code>dimnames</code> of <code>x</code> .

`one.d.array` (Optional, defaults to `FALSE`. If `TRUE`, a one-dimensional array result will be an object with a `dim` attribute of length 1, and possibly a `dimnames` attribute. If `FALSE`, a one-dimensional result will be a vector object (named if `named.vector==TRUE`).

Details

Dimensions can only be dropped if their extent is one, i.e., dimension `i` of array `x` can be dropped only if `dim(x)[i]==1`. It is an error to request `adrop` to drop a dimension whose extent is not 1.

Value

If `x` is an object with a `dim` attribute (e.g., a matrix or array), then `adrop` returns an object like `x`, but with the requested extents of length one removed. Any accompanying `dimnames` attribute is adjusted and returned with `x`.

Author(s)

Tony Plate <tplate@acm.org>

See Also

[abind](#)

Examples

```
x <- array(1:24,dim=c(2,3,4),dimnames=list(letters[1:2],LETTERS[1:3],letters[23:26]))
adrop(x[,,,drop=FALSE],drop=1)
adrop(x[,1,,drop=FALSE],drop=2)
adrop(x[, ,1,drop=FALSE],drop=3)
adrop(x[1,1,1,drop=FALSE],drop=1)
adrop(x[1,1,1,drop=FALSE],drop=2)
adrop(x[1,1,1,drop=FALSE],drop=3)
adrop(x[1,1,1,drop=FALSE],drop=1:2)
adrop(x[1,1,1,drop=FALSE],drop=1:2,one.d=TRUE)
adrop(x[1,1,1,drop=FALSE],drop=1:2,named=FALSE)
dim(adrop(x[1,1,1,drop=FALSE],drop=1:2,one.d=TRUE))
dimnames(adrop(x[1,1,1,drop=FALSE],drop=1:2,one.d=TRUE))
names(adrop(x[1,1,1,drop=FALSE],drop=1:2,one.d=TRUE))
dim(adrop(x[1,1,1,drop=FALSE],drop=1:2))
dimnames(adrop(x[1,1,1,drop=FALSE],drop=1:2))
names(adrop(x[1,1,1,drop=FALSE],drop=1:2))
```

Index

*Topic **array**

abind, 1

adrop, 4

*Topic **manip**

abind, 1

adrop, 4

abind, 1, 5

adrop, 4