Matrices

Description

`matrix` creates a matrix from the given set of values.

`as.matrix` attempts to turn its argument into a matrix.

`is.matrix` tests if its argument is a (strict) matrix.

Usage

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,
       dimnames = NULL)

as.matrix(x, ...)
## S3 method for class 'data.frame'
as.matrix(x, rownames.force = NA, ...)

is.matrix(x)
```

Arguments

`data`	an optional data vector (including a list or `expression` vector). Non-atomic classed `R` objects are coerced by `as.vector` and all attributes discarded.
`nrow`	the desired number of rows.
`ncol`	the desired number of columns.
`byrow`	logical. If `FALSE` (the default) the matrix is filled by columns, otherwise the matrix is filled by rows.
`dimnames`	A `dimnames` attribute for the matrix: `NULL` or a `list` of length 2 giving the row and column names respectively. An empty list is treated as `NULL`, and a list of length one as row names. The list can be named, and the list names will be used as names for the dimensions.
`x`	an `R` object.
`...`	additional arguments to be passed to or from methods.
`rownames.force`	logical indicating if the resulting matrix should have character (rather than `NULL`) `rownames`. The default, `NA`, uses `NULL` rownames if the data frame has 'automatic' row.names or for a zero-row data frame.

Details

If one of `nrow` or `ncol` is not given, an attempt is made to infer it from the length of `data` and the other parameter. If neither is given, a one-column matrix is returned.

If there are too few elements in `data` to fill the matrix, then the elements in `data` are recycled. If `data` has length zero, `NA` of an appropriate type is used for atomic vectors (`0` for raw vectors) and `NULL` for lists.

`is.matrix` returns `TRUE` if `x` is a vector and has a "`dim`" attribute of length 2) and `FALSE` otherwise. Note that a `data.frame` is **not** a matrix by this test. The function is generic: you can write methods to handle specific classes of objects, see InternalMethods.

`as.matrix` is a generic function. The method for data frames will return a character matrix if there is only atomic columns and any non-(numeric/logical/complex) column, applying `as.vector` to factors and `format` to other non-character columns. Otherwise, the usual coercion hierarchy (logical < integer < double < complex) will be used, e.g., all-logical data frames will be coerced to a logical matrix, mixed logical-integer will give a integer matrix, etc.

The default method for `as.matrix` calls `as.vector(x)`, and hence e.g. coerces factors to character vectors.

When coercing a vector, it produces a one-column matrix, and promotes the names (if any) of the vector to the rownames of the matrix.

`is.matrix` is a primitive function.

The `print` method for a matrix gives a rectangular layout with dimnames or indices. For a list matrix, the entries of length not one are printed in the form `integer,7` indicating the type and length.

Note

If you just want to convert a vector to a matrix, something like

```
  dim(x) <- c(nx, ny)
  dimnames(x) <- list(row_names, col_names)
```

will avoid duplicating `x`.

References

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

See Also

`data.matrix`, which attempts to convert to a numeric matrix.

A matrix is the special case of a two-dimensional array.

Examples

```
is.matrix(as.matrix(1:10))
!is.matrix(warpbreaks)  # data.frame, NOT matrix!
warpbreaks[1:10,]
as.matrix(warpbreaks[1:10,])  # using as.matrix.data.frame(.) method

## Example of setting row and column names
mdat <- matrix(c(1,2,3, 11,12,13), nrow = 2, ncol = 3, byrow = TRUE,
               dimnames = list(c("row1", "row2"),
                               c("C.1", "C.2", "C.3")))
mdat
```
