sparse.model.matrix {Matrix} R Documentation

Construct Sparse Design / Model Matrices

Description

Construct a sparse model or “design” matrix, form a formula and data frame (`sparse.model.matrix`) or a single factor (`fac2sparse`).

The `fac2[Ss]parse()` functions are utilities, also used internally in the principal user level function `sparse.model.matrix()`.

Usage

```
sparse.model.matrix(object, data = environment(object),
                    contrasts.arg = NULL, xlev = NULL, transpose = FALSE,
                    drop.unused.levels = FALSE, row.names = TRUE,
                    verbose = FALSE, ...)

fac2sparse(from, to = c("d", "i", "l", "n", "z"),
           drop.unused.levels = TRUE, giveCsparse = TRUE)
fac2Sparse(from, to = c("d", "i", "l", "n", "z"),
           drop.unused.levels = TRUE, giveCsparse = TRUE,
           factorPatt12, contrasts.arg = NULL)
```

Arguments

`object`

an object of an appropriate class. For the default method, a model formula or terms object.

`data`

a data frame created with `model.frame`. If another sort of object, `model.frame` is called first.

`contrasts.arg`

for `sparse.model.matrix()`:

> A list, whose entries are contrasts suitable for input to the `contrasts` replacement function and whose names are the names of columns of `data` containing `factor`s.

for `fac2Sparse()`:

> character string or `NULL` or (coercable to) `"sparseMatrix"`, specifying the contrasts to be applied to the factor levels.

`xlev`

to be used as argument of `model.frame` if `data` has no `"terms"` attribute.

`transpose`

logical indicating if the *transpose* should be returned; if the transposed is used anyway, setting `transpose = TRUE` is more efficient.

`drop.unused.levels`

should factors have unused levels dropped? The default for

	`sparse.model.matrix` has been changed to `FALSE`, 2010-07, for compatibility with R's standard (dense) `model.matrix()`.
`row.names`	logical indicating if row names should be used.
`verbose`	logical or integer indicating if (and how much) progress output should be printed.
`...`	further arguments passed to or from other methods.
`from`	(for `fac2sparse()`:) a `factor`.
`to`	a character indicating the "kind" of sparse matrix to be returned. The default, `"d"` is for `double`.
`giveCsparse`	(for `fac2sparse()`:) logical indicating if the result must be a `CsparseMatrix`.
`factorPatt12`	logical vector, say `fp`, of length two; when `fp[1]` is true, return "contrasted" `t(X)`; when `fp[2]` is true, the original ("dummy") `t(X)`, i.e, the result of `fac2sparse()`.

Value

a sparse matrix, extending `CsparseMatrix` (for `fac2sparse()` if `giveCsparse` is true as per default; a `TsparseMatrix`, otherwise).

For `fac2Sparse()`, a `list` of length two, both components with the corresponding transposed model matrix, where the corresponding `factorPatt12` is true.

Note that `model.Matrix(*, sparse=TRUE)` from package **MatrixModels** may be often be preferable to `sparse.model.matrix()` nowadays, as `model.Matrix()` returns `modelMatrix` objects with additional slots `assign` and `contrasts` which relate back to the variables used.

`fac2sparse()`, the basic workhorse of `sparse.model.matrix()`, returns the *transpose* (`t`) of the model matrix.

Author(s)

Doug Bates and Martin Maechler, with initial suggestions from Tim Hesterberg.

See Also

`model.matrix` in standard R's package **stats**.
`model.Matrix` which calls `sparse.model.matrix` or `model.matrix` depending on its `sparse` argument may be preferred to `sparse.model.matrix`.

`as(f, "sparseMatrix")` (see `coerce(from = "factor", ..)` in the class doc sparseMatrix) produces the *transposed* sparse model matrix for a single factor `f` (and *no* contrasts).

Examples

```
dd <- data.frame(a = gl(3,4), b = gl(4,1,12))# balanced 2-way
options("contrasts") # the default:  "contr.treatment"
sparse.model.matrix(~ a + b, dd)
sparse.model.matrix(~ -1+ a + b, dd)# no intercept --> even sparser
sparse.model.matrix(~ a + b, dd, contrasts = list(a="contr.sum"))
sparse.model.matrix(~ a + b, dd, contrasts = list(b="contr.SAS"))

## Sparse method is equivalent to the traditional one :
stopifnot(all(sparse.model.matrix(~ a + b, dd) ==
              Matrix(model.matrix(~ a + b, dd), sparse=TRUE)),
          all(sparse.model.matrix(~ 0+ a + b, dd) ==
              Matrix(model.matrix(~ 0+ a + b, dd), sparse=TRUE)))


(ff <- gl(3,4,, c("X","Y", "Z")))
fac2sparse(ff) #  3 x 12 sparse Matrix of class "dgCMatrix"
##
##  X  1 1 1 1 . . . . . . . .
##  Y  . . . . 1 1 1 1 . . . .
##  Z  . . . . . . . . 1 1 1 1

## can also be computed via sparse.model.matrix():
f30 <- gl(3,0    )
f12 <- gl(3,0, 12)
stopifnot(
  all.equal(t( fac2sparse(ff) ),
            sparse.model.matrix(~ 0+ff),
            tolerance = 0, check.attributes=FALSE),
  is(M <- fac2sparse(f30, drop= TRUE),"CsparseMatrix"), dim(M) == c(0, 0),
  is(M <- fac2sparse(f30, drop=FALSE),"CsparseMatrix"), dim(M) == c(3, 0),
  is(M <- fac2sparse(f12, drop= TRUE),"CsparseMatrix"), dim(M) == c(0,12),
  is(M <- fac2sparse(f12, drop=FALSE),"CsparseMatrix"), dim(M) == c(3,12)
 )
```

[Package *Matrix* version 1.2-5 Index]