

# Package: INLAtools (via r-universe)

June 4, 2026

**Type** Package

**Title** Functionalities for the 'INLA' Package

**Version** 0.1.4.901

**Maintainer** Elias Teixeira Krainski <elias.krainski@kaust.edu.sa>

**Description** Contain code to work with a C struct, in short cgeneric, to define a Gaussian Markov random (GMRF) model. The cgeneric contain code to specify GMRF elements such as the graph and the precision matrix, and also the initial and prior for its parameters, useful for model inference. It can be accessed from a C program and is the recommended way to implement new GMRF models in the 'INLA' package (<<https://www.r-inla.org>>). The 'INLAtools' implement functions to evaluate each one of the model specifications from R. The implemented functionalities leverage the use of 'cgeneric' models and provide a way to debug the code as well to work with the prior for the model parameters and to sample from it. The `generic0` can be used to implement intrinsic models with the scaling as proposed in Sørbye & Rue (2014) <[doi:10.1016/j.spasta.2013.06.004](https://doi.org/10.1016/j.spasta.2013.06.004)>, and the required constraints. A very useful functionality is the Kronecker product method that creates a new model from multiple cgeneric models. It also works with the rgeneric, the R version of the cgeneric intended to easy try implementation of new GMRF models. The Kronecker between two cgeneric models where each one needs a constraint, such as spatio-temporal intrinsic interaction models, the needed constraints are automatically set.

**License** GPL (>=2)

**URL** <https://github.com/eliaskrainski/INLAtools>

**BugReports** <https://github.com/eliaskrainski/INLAtools/issues>

**Depends** R (>= 4.3), Matrix

**Imports** methods, utils

**Encoding** UTF-8

**LazyData** true  
**LazyDataCompression** xz  
**NeedsCompilation** yes  
**Roxygen** list(markdown = TRUE)  
**Config/roxygen2/version** 8.0.0  
**Repository** https://eliaskrainski.r-universe.dev  
**Date/Publication** 2026-06-04 06:30:29 UTC  
**RemoteUrl** https://github.com/eliaskrainski/INLAtools  
**RemoteRef** HEAD  
**RemoteSha** 20f061b2c55a4b412236138422d1ea9863c3745b

## Contents

cgeneric-class . . . . .	2
cgeneric_generic0 . . . . .	5
cgeneric_get . . . . .	6
extraconstr . . . . .	8
findGetFunction . . . . .	9
inlaQ . . . . .	9
is.zero . . . . .	10
kronecker . . . . .	10
multi_generic_model . . . . .	11
packageCheck . . . . .	12
pc-utils . . . . .	13
rgeneric-class . . . . .	14
rgeneric_get . . . . .	15
Sparse . . . . .	17
upperPadding . . . . .	18
<b>Index</b>	<b>20</b>

---

cgeneric-class	<i>Organize data for the latent GMRF C interface for INLA.</i>
----------------	--

---

## Description

A GMRF is defined from model parameters  $\theta$  that would parametrize a (sparse) precision matrix.

The elements of a GMR are:

- graph to define the non-zero precision matrix pattern. only the upper triangle including the diagonal is needed.
- Q vector where the
  - first element (N) is the size of the matrix,

- second element (M) is the number of non-zero elements in the upper part (including) diagonal
- the remaining (M) elements are the actual precision (upper triangle plus diagonal) elements whose order shall follow the graph definition.
- `mu` the mean vector,
- initial vector with
  - first element as the number of the parameters in the model
  - remaining elements should be the initials for the model parameters.
- `log.norm.const` log of the normalizing constant.
- `log.prior` log of the prior for the model parameters.

### Usage

```

cgeneric(model, ...)

## S3 method for class 'character'
cgeneric(model, ...)

## S3 method for class '`function`'
cgeneric(model, ...)

## S3 method for class 'cgeneric'
cgeneric(model, ...)

## S3 method for class 'inla.cgeneric'
cgeneric(model, ...)

cgenericBuilder(...)

mapper1(model)

cgeneric_shlib_path(package, useINLApcomp, debug)

## S3 method for class 'cgeneric'
print(x, ...)

## S3 method for class 'cgeneric'
summary(object, ...)

## S3 method for class 'cgeneric'
plot(x, y, ...)

```

### Arguments

`model` object class for what a `cgeneric` method exists. if it is a character, a specific function will be called, for example `cgeneric("iid", ...)` calls `cgeneric_iid(...)`.

...	additional arguments passed on to methods It should include <code>n</code> and <code>debug</code> . For <code>cgenericBuild</code> and <code>model</code> as a character string with the name of the C function and <code>shlib</code> as the path to the shared object containing such function. If <code>shlib</code> is not provided it can be built using <code>cgeneric_shlib_path</code> for the package (character with the R package containing it), in which <code>useINLAprecomp</code> (logical) indicates if it is to use the one contained within INLA or in the package. When using the <code>inlabru</code> package, one can also provide a <code>mapper</code> which will be evaluated by the <code>bru_get_mapper</code> in <code>inlabru</code> 's function.
<code>package</code>	character giving the name of the package that contains the <code>cgeneric</code> model.
<code>useINLAprecomp</code>	logical, indicating if it is to use the shared object previously copied and compiled by INLA.
<code>debug</code>	integer, used as <code>verbose</code> in <code>debug</code> .
<code>x</code>	a <code>cgeneric</code> object
<code>object</code>	a <code>cgeneric</code> object
<code>y</code>	not used

### Value

a method to build a `cgeneric` should return a named list of `cgeneric` class that contains a named list `f` that contains (at least):

- `model` a character always equal to `cgeneric`,
- `n` an integer greater than 0, and
- `cgeneric` as a named list that contains the data needed to define the model. Each element on `...$f$cgeneric` is also a named list containing ints, doubles, characters, matrices and smatrices.
- (possible) `extraconstr` as a named list with: `A` as a `n` times `k` matrix and `e` as a length `k` vector.

The `cgeneric_shlib_path` function returns a character with the path to the shared lib.

### Functions

- `cgeneric(character)`: Call a function named as `cgeneric_[model]` to build a `cgeneric`.
- `cgeneric(`function`)`: The `cgeneric` method for `function`.
- `cgeneric(cgeneric)`: Check, append `cgeneric` class, returns `model` unchanged.
- `cgeneric(inla.cgeneric)`: Check and converts a regular `inla.cgeneric` object to `cgeneric`.
- `cgenericBuilder()`: Build a `cgeneric` from a list of arguments.
- `mapper1()`: A default mapper for a `cgeneric/rgeneric` model
- `cgeneric_shlib_path()`: Make the lib path for the shared lib of a package.
- `print(cgeneric)`: Print the `cgeneric` object
- `summary(cgeneric)`: A summary for a `cgeneric` object
- `plot(cgeneric)`: A plot for a `cgeneric` object

**Note**

The graph and Q non-zero pattern should match, its elements should be ordered by row, and only its upper part stored.

**Examples**

```
cg <- cgeneric("iid", n = 10, param = c(1, 0.05))
cg
```

---

`cgeneric_generic0`      *Build a cgeneric object for a generic0 model.*

---

**Description**

Build data needed to implement a model whose precision has a conditional precision parameter. This uses the C interface in the 'INLA' package, that can be used as a linear predictor model component with an 'f' term.

**Usage**

```
cgeneric_generic0(R, param, constr = TRUE, scale = TRUE, ...)
cgeneric_iid(n, param, constr = FALSE, ...)
```

**Arguments**

R	the structure matrix for the model definition.
param	length two vector with the parameters a and p for the PC-prior distribution defined from $P(\sigma > a) = p$ where $\sigma$ can be interpreted as marginal standard deviation of the process if scale = TRUE. See details.
constr	logical indicating if it is to add a sum-to-zero constraint. Default is TRUE.
scale	logical indicating if it is to scale the model. See details.
...	arguments (debug,useINLAprecomp,shlib) passed on to cgeneric.
n	integer required to specify the model size

**Details**

The precision matrix is defined as

$$Q = \tau R$$

where the structure matrix R is supplied by the user and  $\tau$  is the precision parameter. Following Sørbye and Rue (2014), if scale = TRUE the model is scaled so that

$$Q = \tau s R$$

where  $s$  is the geometric mean of the diagonal elements of the generalized inverse of  $R$ .

$$s = \exp \sum_i \log((R^-)_{ii})/n$$

If the model is scaled, the geometric mean of the marginal variances, the diagonal of  $Q^{-1}$ , is one. Therefore, when the model is scaled,  $\tau$  is the marginal precision, otherwise  $\tau$  is the conditional precision.

### Value

a cgeneric object, see `cgeneric-class()`.

### Functions

- `cgeneric_iid()`: The `cgeneric_iid` uses the `cgeneric_generic0` with the structure matrix as the identity.

### References

Sigrunn Holbek Sørbye and Håvard Rue (2014). Scaling intrinsic Gaussian Markov random field priors in spatial modelling. *Spatial Statistics*, vol. 8, p. 39-51.

---

<code>cgeneric_get</code>	<i>cgeneric_get is an internal function used to query graph, Q, initial, mu or log_prior from a cgeneric model.</i>
---------------------------	---

---

### Description

The `generic_get` retrieve a model property specified by `cmd` on an `cgeneric` object. The functions listed below are for each `cmd` case.

### Usage

```
cgeneric_get(
  model,
  cmd = c("graph", "Q", "initial", "mu", "log_prior"),
  theta,
  optimize = TRUE
)
```

```
cgeneric_initial(model)
```

```
cgeneric_mu(model, theta)
```

```
cgeneric_graph(model, optimize)
```

```
cgeneric_Q(model, theta, optimize)
```

```
cgeneric_prior(model, theta)
```

**Arguments**

model	a cgeneric object.
cmd	an string to specify which model element to get
theta	numeric vector with the model parameters. If missing, the initial will be used.
optimize	logical indicating if it is to be returned only the elements and not as a sparse matrix.

**Value**

depends on cmd

numeric scalar (if numeric vector is provided for theta) or vector (if numeric matrix is provided for theta).

**Functions**

- `cgeneric_initial()`: Retrieve the initial parameter(s) of an cgeneric model.
- `cgeneric_mu()`: Evaluate the mean for an cgeneric model.
- `cgeneric_graph()`: Retrieve the graph of an cgeneric object
- `cgeneric_Q()`: Retrieve the precision of an cgeneric object
- `cgeneric_prior()`: Evaluate the prior for an cgeneric model

**See Also**

check the examples in `cgeneric_generic0()`

**Examples**

```
library(INLAtools)
old.par <- par(no.readonly = TRUE)

## Setting the prior parameters
prior.par <- c(1, 0.5) # P(sigma > 1) = 0.5
cmodel <- cgeneric(
  model = "iid", n = 10,
  param = prior.par)

## prior summaries: sigma and log-precision
(lamb <- -log(prior.par[2])/prior.par[1])
(smedian <- qexp(0.5, lamb))
(smean <- 1/lamb)

## mode: at the minimum of - log-prior
(lpmode <- optimize(function(x)
  -cgeneric_prior(cmodel, theta = x),
  c(-10, 30))$minimum)
## mean: integral of x*f(x)dx
(lpmean <- integrate(function(x)
  exp(cgeneric_prior(cmodel, theta = matrix(x, 1)))*x,
```

```

-10, 30)$value)

## prior visualization: log(precision) and sigma
par(mfrow = c(1, 2))
plot(function(x)
  exp(cgeneric_prior(cmodel, theta = matrix(x, nrow=1))),
  -3, 3, n = 601, xlab = "log-precision",
  ylab = "density")
abline(v = lpmode, lwd = 3, col = 2)
rug(-2*log(smmedian), lwd = 3, col = 3)
rug(lpmean, lwd = 3, col = 4)
plot(function(x)
  exp(cgeneric_prior(cmodel,
    theta = matrix(
      -2*log(x),
      nrow = 1))+log(2)-log(x)),
  1/100, 10, n = 1000,
  xlab = expression(sigma),
  ylab = "density")
plot(function(x) dexp(x, lamb),
  1/100, 10, n = 1000,
  add = TRUE, lty = 2, col = 2)
rug(smmedian, lwd = 3, col = 3)
rug(smmean, lwd = 3, col = 4)

par(old.par)

```

---

extraconstr                      *Kronecker (product) between extraconstr, implemented for [kronecker\(\)](#) methods.*

---

## Description

Kronecker (product) between extraconstr, implemented for [kronecker\(\)](#) methods.

## Usage

```
kronecker_extraconstr(c1, c2, n1, n2)
```

## Arguments

c1, c2	named list with two elements: A and e, where nrow(A) should be equal to length(e). These are constraint definitions.
n1, n2	integer with each model's length.

## Value

The constraint definition for the whole latent model built from the Kronecker product. A length two named list. 'A' a matrix and 'e' a vector where nrow(A)=length(e) and ncol(A)=(n1\*n2).

---

findGetFunction	<i>Search a function and retrieve it.</i>
-----------------	---

---

**Description**

Search a function and retrieve it.

**Usage**

```
findGetFunction(fName, package, debug = FALSE)
```

**Arguments**

fName	character with the name of the function
package	character with the package name
debug	logical indicating if it is to print intermediate progress finding

**Details**

if 'missing(package)' it will search on the loaded packages, first in the exported functions, and then among the non-exported ones. NOTE: 'package' can include any installed package, see [installed.packages\(\)](#)

**Value**

function. The (first) package name where it was found is returned as an attribute named "package"

---

inlaQ	<i>Define the method to extract the precision from an inla output object.</i>
-------	---

---

**Description**

Define the method to extract the precision from an inla output object.

**Usage**

```
inlaQ(model, ...)
```

**Arguments**

model	an inla output
...	used to pass the 'prior' argument, as logical (default is TRUE) to indicate if it is to retrieve the prior or the posterior precision.

**Details**

extract the joint prior precision for the latent field at the mode of the hyperparameters

---

is.zero	<i>Define the is.zero method</i>
---------	----------------------------------

---

**Description**

Define the is.zero method

**Usage**

```
is.zero(x, tol)

## Default S3 method:
is.zero(x, tol)

## S3 method for class 'matrix'
is.zero(x, tol)

## S3 method for class 'Matrix'
is.zero(x, tol)
```

**Arguments**

x	an R object
tol	numeric to be used as (absolute) tolerance. if missing (default) it will consider $x==0$ .

**Value**

logical

**Methods (by class)**

- `is.zero(default)`: The `is.zero.default` definition
- `is.zero(matrix)`: The `is.zero.matrix` definition
- `is.zero(Matrix)`: The `is.zero.Matrix` definition

---

kronecker	<i>Kronecker between cgenericlrgeneric to implement interaction between GMRF models.</i>
-----------	--

---

**Description**

Kronecker between cgenericlrgeneric to implement interaction between GMRF models.

**Usage**

```
## S4 method for signature 'cgeneric,cgeneric'
kronecker(X, Y, FUN = "*", make.dimnames = FALSE, ...)

## S4 method for signature 'cgeneric,rgeneric'
kronecker(X, Y, FUN = "*", make.dimnames = FALSE, ...)

## S4 method for signature 'rgeneric,cgeneric'
kronecker(X, Y, FUN = "*", make.dimnames = FALSE, ...)

## S4 method for signature 'rgeneric,rgeneric'
kronecker(X, Y, FUN = "*", make.dimnames = FALSE, ...)
```

**Arguments**

X	cgeneric or rgeneric
Y	cgeneric or rgeneric
FUN	see <a href="#">kronecker()</a>
make.dimnames	see <a href="#">kronecker()</a>
...	see <a href="#">kronecker()</a>

**Value**

A cgeneric object if kronecker between two cgeneric, otherwise a rgeneric object.

**Functions**

- `kronecker(X = cgeneric, Y = cgeneric)`: Build a cgeneric to implement the interaction between two GMRF models each one implemented as a cgeneric.
- `kronecker(X = cgeneric, Y = rgeneric)`: Build a rgeneric to implement the interaction between two GMRF models, the first as a cgeneric and the second as a rgeneric.
- `kronecker(X = rgeneric, Y = cgeneric)`: Build a rgeneric to implement the interaction between two GMRF models, the first as a rgeneric and the second as a cgeneric.
- `kronecker(X = rgeneric, Y = rgeneric)`: Build a rgeneric to implement the interaction between two GMRF models each one implemented as a rgeneric.

---

multi\_generic\_model    *Combine two or more cgeneric or rgeneric models*

---

**Description**

Constructs a multiple kronecker product model from a list of model objects. The resulting model contains a corresponding `inlabru::bm_multi()` mapper. This can be used as an alternative to a binary tree of kronecker product models.

**Usage**

```
multi_generic_model(models, ...)

multi_generic_model_mapper(models)
```

**Arguments**

models            A list of cgeneric or rgeneric models, optionally with names  
 ...               Arguments passed on to every kronecker() call.

**Details**

The last model in the list has the slowest index variation, and the first model has the fastest index variation. This matches the latent variable ordering of standard INLA: f() model components with (main, group, replicate).

**Value**

A 'cgeneric' or 'rgeneric' model object, containing a multi-kronecker product model, with a corresponding `inlabru::bm_multi()` mapper.

**Functions**

- `multi_generic_model_mapper()`: Build the `bm_multi` mapper for a list of models

---

packageCheck	<i>To check package version and load</i>
--------------	--

---

**Description**

To check package version and load

**Usage**

```
packageCheck(name, minimum_version, quietly = FALSE)
```

**Arguments**

name                character with the name of the package  
 minimum\_version    character with the minimum required version  
 quietly            logical indicating if messages shall be printed

**Note**

Original in inlabru package function `check_package_version_and_load`

---

 pc-utils

---

*Internal functions to check PC-prior parameters.*


---

**Description**

Internal functions to check PC-prior parameters.

**Usage**

```
pcParamCheck(npars, reference, probability)
```

```
pcLrange(lrange, lam, d = 2, log = FALSE)
```

```
pcrange(range, lam, d = 2, log = FALSE)
```

**Arguments**

npars	number of parameters.
reference	numeric vector to set the reference for each parameter for its PC-prior.
probability	numeric vector with to set the probability statement of the PC prior for each parameter. For sigma probability statement is $P(\text{sigma} > \text{reference}) = p$ whereas for range it is $P(\text{range} < \text{reference})$ . If NA, 0 or 1, the corresponding reference will be used as fixed. If missing, all the parameters considered as known (fixed) and equal the corresponding reference value.
lrange	numeric with the log of the (practical) range
lam	numeric with the prior parameter
d	integer to specify the domain dimension
log	logical indicating if the density is to be returned in the log scale
range	numeric with the of the (practical) range.

**Functions**

- `pcParamCheck()`: Check the PC-prior arguments.
- `pcLrange()`: Penalized Complexity (PC) prior for the log of the practical range.
- `pcrange()`: Penalized Complexity (PC) prior for the practical range.

**Examples**

```
# P(range < 2.0) = 0.1
lam <- -log(0.1) * 2.0
plot(function(x) pcrange(x, lam), 1/100, 10, n = 100)
```

---

 rgeneric-class

 Organize data for the latent GMRF R interface for INLA.
 

---

## Description

A GMRF is defined from model parameters  $\theta$  that would parametrize a (sparse) precision matrix.

The elements of a GMR are:

- graph to define the non-zero precision matrix pattern. only the upper triangle including the diagonal is needed.
- Q vector where the
  - first element (N) is the size of the matrix,
  - second element (M) is the number of non-zero elements in the upper part (including) diagonal
  - the remaining (M) elements are the actual precision (upper triangle plus diagonal) elements whose order shall follow the graph definition.
- mu the mean vector,
- initial vector with
  - first element as the number of the parameters in the model
  - remaining elements should be the initials for the model parameters.
- log.norm.const log of the normalizing constant.
- log.prior log of the prior for the model parameters.

## Usage

```
rgeneric(model, n, debug = FALSE, ...)
```

```
## Default S3 method:
```

```
rgeneric(model, n, debug = FALSE, ...)
```

```
## S3 method for class 'rgeneric'
```

```
rgeneric(model, ...)
```

```
## S3 method for class 'inla.rgeneric'
```

```
rgeneric(model, ...)
```

```
## S3 method for class 'rgeneric'
```

```
print(x, ...)
```

```
## S3 method for class 'rgeneric'
```

```
summary(object, ...)
```

```
## S3 method for class 'rgeneric'
```

```
plot(x, y, ...)
```

**Arguments**

model	an object used to define the model. See the 'rgeneric' vignette from the INLA package.
n	integer with the dimension of the model
debug	logical indicating debug state.
...	not used
x	a rgeneric object
object	a rgeneric object
y	not used

**Value**

rgeneric/ inla.rgeneric object.

**Functions**

- `rgeneric(default)`: The rgeneric default method.
- `rgeneric(rgeneric)`: Returns the model object unchanged.
- `rgeneric(inla.rgeneric)`: Check and converts a regular `inla.rgeneric` object to `rgeneric`.
- `print(rgeneric)`: Print the rgeneric object
- `summary(rgeneric)`: A summary for a rgeneric object
- `plot(rgeneric)`: A plot for a rgeneric object

**Note**

Recommended for prototyping, whereas `cgeneric` is recommended for production.

---

<code>rgeneric_get</code>	<i>rgeneric_get is an internal function used to query graph, Q, initial, mu or prior from a rgeneric.</i>
---------------------------	---

---

**Description**

The `generic_get` retrieve a model property specified by `cmd` on an `rgeneric` object. The functions listed below are for each `cmd` case.

**Usage**

```

rgeneric_get(
  model,
  cmd = c("graph", "Q", "initial", "mu", "log_prior"),
  theta,
  ...
)

rgeneric_initial(model)

rgeneric_mu(model, theta)

rgeneric_graph(model, optimize)

rgeneric_Q(model, theta, optimize)

rgeneric_prior(model, theta)

```

**Arguments**

model	a rgeneric object.
cmd	an string to specify which model element to get
theta	numeric vector with the model parameters. If missing, the initial will be used.
...	additional arguments passed on to methods. E.g.: optimize = FALSE return the graph and precision as a sparse matrix whereas optimize = TRUE retur the graph as arow/col indexes and the precision as a numeric vector with its elements.
optimize	logical indicating if it is to be returned only the elements and not as a sparse matrix.

**Value**

depends on cmd

numeric scalar (if numeric vector is provided for theta) or vector (if numeric matrix is provided for theta).

**Functions**

- `rgeneric_initial()`: Retrieve the initial parameter(s) of an rgeneric model.
- `rgeneric_mu()`: Evaluate the mean for an rgeneric model.
- `rgeneric_graph()`: Retrieve the graph of an rgeneric object
- `rgeneric_Q()`: Retrieve the precision of an rgeneric object
- `rgeneric_prior()`: Evaluate the prior for an rgeneric model

**Examples**

```

library(INLAtools)
old.par <- par(no.readonly = TRUE)

## Setting the prior parameters
prior.par <- c(1, 0.5) # P(sigma > 1) = 0.5
cmodel <- cgeneric(
  model = "iid", n = 10,
  param = prior.par)

## prior summaries: sigma and log-precision
(lamb <- -log(prior.par[2])/prior.par[1])
(smmedian <- qexp(0.5, lamb))
(smmean <- 1/lamb)

## mode: at the minimum of - log-prior
(lpmode <- optimize(function(x)
  -cgeneric_prior(cmodel, theta = x),
  c(-10, 30))$minimum)
## mean: integral of x*f(x)dx
(lpmean <- integrate(function(x)
  exp(cgeneric_prior(cmodel, theta = matrix(x, 1)))*x,
  -10, 30)$value)

## prior visualization: log(precision) and sigma
par(mfrow = c(1, 2))
plot(function(x)
  exp(cgeneric_prior(cmodel, theta = matrix(x, nrow=1))),
  -3, 3, n = 601, xlab = "log-precision",
  ylab = "density")
abline(v = lpmode, lwd = 3, col = 2)
rug(-2*log(smmedian), lwd = 3, col = 3)
rug(lpmean, lwd = 3, col = 4)
plot(function(x)
  exp(cgeneric_prior(cmodel,
  theta = matrix(
    -2*log(x),
    nrow = 1))+log(2)-log(x)),
  1/100, 10, n = 1000,
  xlab = expression(sigma),
  ylab = "density")
plot(function(x) dexp(x, lamb),
  1/100, 10, n = 1000,
  add = TRUE, lty = 2, col = 2)
rug(smmedian, lwd = 3, col = 3)
rug(smmean, lwd = 3, col = 4)

par(old.par)

```

**Description**

To store in i,j,x sparse matrix format

**Usage**

```
Sparse(A, unique = TRUE, na.rm = FALSE, zeros.rm = FALSE)
```

**Arguments**

A	matrix or Matrix
unique	logical (default is TRUE) to ensure that the internal representation is unique and there are no duplicated entries. (Do not change this unless you know what you are doing.)
na.rm	logical (default is FALSE) indicating if it is to replace 'NA's in the matrix with zeros.
zeros.rm	logical (default is FALSE) indicating if it is to remove zeros in the matrix. Applied after na.rm.

**Note**

This is based in INLA::inla.as.sparse(), but allow all combinations of 'na.rm' and 'zeros.rm'.

---

upperPadding

*Padding (a list of) sparse matrices.*

---

**Description**

Padding (a list of) sparse matrices.

**Usage**

```
upperPadding(M, relative = FALSE, ...)
```

**Arguments**

M	'Matrix' (or a list of them).
relative	logical. If 'M' is a list, it indicates if it is to be returned a relative index and the value for each matrix. See details.
...	additional arguments passed to <a href="#">Sparse</a> .

**Details**

This is useful to prepare a matrix, or a list of, sparse matrices for use in some 'cgeneric' code.

Define a graph of the union of the supplied matrices and return the row ordered diagonal plus upper triangle after padding with zeroes each one so that all the returned matrices have the same pattern.

If `relative=FALSE`, each columns of 'xx' is the elements of the corresponding matrix after being padded to fill the pattern of the union graph. If `relative=TRUE`, each element of 'xx' would be a list with a relative index, 'r', for each non-zero elements of each matrix is returned relative to the union graph, the non-lower elements, 'x', of the corresponding matrix, and a vector, 'o', with the number of non-zero elements for each line of each resulting matrix.

**Value**

If a unique matrix is given, return the upper triangle considering the 'T' representation in the `dgTMatrix`, from the `Matrix` package. If a list of matrices is given, return a list of two elements: 'graph' and 'xx'. The 'graph' is the union of the graph from each matrix. If `relative=FALSE`, 'xx' is a matrix with number of column equals the the number of matrices imputed. If `relative=TRUE`, it is a list of length equal the number of matrices imputed. See details.

**Examples**

```
A <- sparseMatrix(
  i = c(1, 1, 2, 3, 3, 5),
  j = c(2, 5, 3, 4, 5, 5),
  x = -c(0:3,NA,1), symmetric = TRUE)
A
upperPadding(A)
upperPadding(A, na.rm = TRUE)
upperPadding(A, zeros.rm = TRUE)
upperPadding(A, na.rm = TRUE, zeros.rm = TRUE)
B <- Diagonal(nrow(A), -colSums(A, na.rm = TRUE))
B
upperPadding(list(a = A, b = B), na.rm = TRUE, zeros.rm = TRUE)
upperPadding(list(a = A, b = B), relative = TRUE)
```

# Index

`cgeneric` (`cgeneric-class`), 2  
`cgeneric-class`, 2  
`cgeneric.cgeneric` (`cgeneric-class`), 2  
`cgeneric.character` (`cgeneric-class`), 2  
`cgeneric.function` (`cgeneric-class`), 2  
`cgeneric.inla.cgeneric`  
    (`cgeneric-class`), 2  
`cgeneric_generic0`, 5, 6  
`cgeneric_generic0()`, 7  
`cgeneric_get`, 6  
`cgeneric_graph` (`cgeneric_get`), 6  
`cgeneric_iid`, 6  
`cgeneric_iid` (`cgeneric_generic0`), 5  
`cgeneric_initial` (`cgeneric_get`), 6  
`cgeneric_mu` (`cgeneric_get`), 6  
`cgeneric_prior` (`cgeneric_get`), 6  
`cgeneric_Q` (`cgeneric_get`), 6  
`cgeneric_shlib_path` (`cgeneric-class`), 2  
`cgenericBuilder` (`cgeneric-class`), 2  
  
`extraconstr`, 8  
  
`findGetFunction`, 9  
  
`inlabru::bm_multi()`, 11, 12  
`inlaQ`, 9  
`installed.packages()`, 9  
`is.zero`, 10  
  
`kronecker`, 10  
`kronecker()`, 8, 11  
`kronecker, cgeneric, cgeneric-method`  
    (`kronecker`), 10  
`kronecker, cgeneric, rgeneric-method`  
    (`kronecker`), 10  
`kronecker, rgeneric, cgeneric-method`  
    (`kronecker`), 10  
`kronecker, rgeneric, rgeneric-method`  
    (`kronecker`), 10  
`kronecker_extraconstr` (`extraconstr`), 8  
  
`mapper1` (`cgeneric-class`), 2  
`multi_generic_model`, 11  
`multi_generic_model_mapper`  
    (`multi_generic_model`), 11  
  
`packageCheck`, 12  
`pc-utils`, 13  
`pcIrange` (`pc-utils`), 13  
`pcParamCheck` (`pc-utils`), 13  
`pcrange` (`pc-utils`), 13  
`plot.cgeneric` (`cgeneric-class`), 2  
`plot.rgeneric` (`rgeneric-class`), 14  
`print.cgeneric` (`cgeneric-class`), 2  
`print.rgeneric` (`rgeneric-class`), 14  
  
`rgeneric` (`rgeneric-class`), 14  
`rgeneric-class`, 14  
`rgeneric.default` (`rgeneric-class`), 14  
`rgeneric.inla.rgeneric`  
    (`rgeneric-class`), 14  
`rgeneric.rgeneric` (`rgeneric-class`), 14  
`rgeneric_get`, 15  
`rgeneric_graph` (`rgeneric_get`), 15  
`rgeneric_initial` (`rgeneric_get`), 15  
`rgeneric_mu` (`rgeneric_get`), 15  
`rgeneric_prior` (`rgeneric_get`), 15  
`rgeneric_Q` (`rgeneric_get`), 15  
  
`Sparse`, 17, 18  
`summary.cgeneric` (`cgeneric-class`), 2  
`summary.rgeneric` (`rgeneric-class`), 14  
  
`upperPadding`, 18