Package 'INLAtools'

June 23, 2025

Type Package

Title Functionalities for the 'INLA' Package

Version 0.0.4

Maintainer Elias Teixeira Krainski <eliaskrainski@gmail.com>

Description Contain code to work with latent Gaussian
Markov random field (GMRF) models. Queries for the
'cgeneric' interface, specified as a way to implement
new GMRF models to be fitted as model components
in the 'INLA' package (<https://www.r-inla.org>).
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.
A Kronecker product method is also implemented to
work with the four possible combinations between
a 'cgeneric' and a 'rgeneric' model.

Additional_repositories https://inla.r-inla-download.org/R/testing

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation yes

Depends R (>= 4.3), Matrix

Imports methods, utils

Suggests INLA (>= 24.02.09)

BuildVignettes true

Author Elias Teixeira Krainski [cre, aut, cph] (ORCID:
<https://orcid.org/0000-0002-7063-2615>),
Haavard Rue' [aut] (ORCID: <https://orcid.org/0000-0002-0222-1881>)

Repository CRAN

Date/Publication 2025-06-23 14:50:05 UTC

Contents

`cgeneric` *Defines a GMRF model to be used with the C interface for* `INLA` *as a latent model.*

Description

This prepare data for the C type to organize data needed for building latent models which are characterized from given model parameters θ and the the following model elements.

- `graph` to define the non-zero precision matrix pattern. only the upper triangle including the diagonal is needed. The order should be by line.
- `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.

See details in `INLA::cgeneric()`

Usage

```
cgeneric(model, ...)
```

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(...), see `cgeneric_iid()` and `cgeneric_generic0()`.
`...`	additional arguments passed on to methods

Value

named list of `cgeneric` class containing the named list `f` that contain `model` (a character always equal to `cgeneric`), `n` (integer) 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`.

See Also

`INLA::cgeneric()` and `methods()`

`cgeneric-class` *The* `cgeneric` *class for* `cgeneric()`.

Description

The `cgeneric` class for `cgeneric()`.

Usage

```
## S3 method for class 'character'
cgeneric(
  model,
  debug = FALSE,
  package,
  useINLAprecomp = TRUE,
  libpath = NULL,
  ...
)

cgeneric_libpath(fName, package, useINLAprecomp = FALSE, debug = FALSE)
```

Arguments

`model`	object class for what a `cgeneric` method exists. E.g., if it is a character, a specific function will be called: cgeneric("iid", ...") calls cgeneric_iid(...)
`debug`	integer, used as verbose in debug.
`package`	character giving the name of the package that contains the `cgeneric` model.
`useINLAprecomp`	logical, indicating if it is to use the shared object previously copied and compiled by INLA.
`libpath`	character, to inform the full path to the shared dynamic library object (this override the arguments `useINLAprecomp` and `package`).
`...`	additional arguments passed to to methods. Some arguments can be used to define specific behavior, such as debug (integer, used as verbose in debug), `useINLAprecomp` (logical, indicating if it is to use the shared object previously copied and compiled by INLA), `package` (character used if `useINLAprecomp` is TRUE, with the package name to build the path) and `libpath` (character, with the path to the shared dynamic library object: this override `useINLAprecomp` and `package`).
`fName`	character with the name of the function used to build the `cgeneric` model.

Details

Method for when `model` is a character. E.g. cgeneric(model = "generic0") calls cgeneric_generic0

Value

character containing the path to the shared lib

Functions

- `cgeneric(character)`: This calls `INLA::inla.cgeneric.define()`
- `cgeneric_libpath()`: Get the shared lib path to use in a `cgeneric` model

`cgeneric_generic0` *Build an* `cgeneric` *object for a* `generic0` *model. See details.*

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 σ 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 detais.
`...`	arguments (debug,useINLAprecomp,libpath) 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 τ is the precision parameter. Following Sørbie 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, τ is the marginal precision, otherwise τ is the conditional precision.

Value

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

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.

See Also

`prior.cgeneric()`

Examples

```
## structured precision matrix model definition
R <- Matrix(toeplitz(c(2,-1,0,0,0)))
R
mR <- cgeneric("generic0", R = R,
  param = c(1, 0.05), scale = FALSE)
graph(mR)
prec(mR, theta = 0)
```

---

cgeneric_get — `cgeneric_get` *is an internal function used by* `graph`, `pred`, `initial`, `mu` *or* `prior` *methods for* `cgeneric`.

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
)

## S3 method for class 'cgeneric'
initial(model)

## S3 method for class 'cgeneric'
mu(model, theta)

## S3 method for class 'cgeneric'
graph(model, optimize)

## S3 method for class 'cgeneric'
prec(model, theta, optimize)

## S3 method for class '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

- `initial(cgeneric)`: Retrive the initial parameter(s) of an `cgeneric` model.
- `mu(cgeneric)`: Evaluate the mean for an `cgeneric` model.
- `graph(cgeneric)`: Retrieve the graph of an `cgeneric` object
- `prec(cgeneric)`: Retrieve the precision of an `cgeneric` object
- `prior(cgeneric)`: Evaluate the prior for an `cgeneric` model

See Also

check the examples in `cgeneric_generic0()`

Examples

```
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)
  -prior(cmodel, theta = x),
  c(-10, 30))$minimum)
## mean: integral of x*f(x)dx
(lpmean <- integrate(function(x)
  exp(prior(cmodel, theta = matrix(x, 1)))*x,
  -10, 30)$value)

## prior visualization: log(precision) and sigma
par(mfrow = c(1, 2))
plot(function(x)
 exp(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(smedian), lwd = 3, col = 3)
rug(lpmean, lwd = 3, col = 4)
plot(function(x)
 exp(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(smedian, lwd = 3, col = 3)
rug(smean, 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)`. This is an extraconstr.
`n1, n2`	integer with each model's length

Value

extraconstr

`findGetFunction` *Search a function and retrieve it.*

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"

`inla.cgeneric.sample` *Draw samples from hyperparameters of a* `cgeneric` *model component from an* `inla` *output, like* `inla::inla.iidkd.sample()`.

Description

Draw samples from hyperparameters of a `cgeneric` model component from an `inla` output, like `inla::inla.iidkd.sample()`.

Usage

```
inla.cgeneric.sample(
  n = 10000,
  result,
  name,
  model,
  from.theta,
  simplify = FALSE
)
```

Arguments

`n`	integer as the sample size.
`result`	an `inla` output.
`name`	character with the name of the model component in the set of random effects.
`model`	a `cgeneric` model
`from.theta`	a function to convert from theta to the desired output for each sample.
`simplify`	logical (see ?sapply).

Value

matrix (if n>1 and length(from.theta)>1) or numeric vector otherwise.

See Also

`prior.cgeneric()`

is.zero *Define the is.zero method*

Description

Define the is.zero method

Usage

```
is.zero(x, ...)

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

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

Arguments

`x`	an R object
`...`	additional arguments

Value

logical

Methods (by class)

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

kronecker — *Kronecker (product) between* cgeneric/rgeneric *models, implemented as* kronecker() *methods.*

Description

Kronecker (product) between cgeneric/rgeneric models, implemented as kronecker() methods.

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 kronecker()
make.dimnames	see kronecker()
...	see kronecker()

Value

if 'X' and 'Y' are cgeneric return a cgeneric, else a rgeneric.

Examples

```
R <- Matrix(crossprod(diff(diag(4))))
m1 <- cgeneric("generic0", R = R, param = c(1, NA),
  scale = FALSE, useINLAprecomp = FALSE)
m2 <- cgeneric("iid", n = 3, param = c(1, 0.5),
  useINLAprecomp = FALSE)
k21 <- kronecker(m2, m1, useINLAprecomp = FALSE)
prec(k21, theta = 0.0)
```

methods — *Methods to work with a* `model`.

Description

For a given model object query the `initial`, `mu`, log `prior`, `graph` or precision `prec` can be evaluated/retrieved.

Usage

```
initial(model)

mu(model, theta)

prior(model, theta)

graph(model, optimize)

prec(model, theta, optimize)

## Default S3 method:
prec(model, ...)

## S4 method for signature 'Matrix'
vcov(object, ...)

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

Arguments

`model`	object to represent a model
`theta`	numeric vector. For `prior` it can be a numeric matrix, with number of lines equal the size of `theta` and each column as a different case.
`optimize`	logical indicating if it is to be returned only the elements and not as a sparse matrix.
`...`	additional arguments passed on
`object`	Matrix supposed to be a sparse precision matrix

Value

the result of the desired query of the 'cgeneric' model. 'graph' and 'prec' can be either a vector (if optimize = TRUE) or a sparse matrix.

Functions

- `initial()`: Retrieve the initial model parameter(s)
- `mu()`: Evaluate the model's mean
- `prior()`: Evaluate the log-prior for a given `theta`
- `graph()`: Retrieve the models' graph
- `prec()`: Retrieve the precision for a given `theta`
- `prec(default)`: The default precision method computes the inverse of the variance
- `vcov(Matrix)`: The `vcov` method for sparse matrices
- `prec(inla)`: Define the prec method for an inla output object

See Also

`prior.cgeneric()`

`packageCheck` *To check package version and load*

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

`rgeneric` *Define rgeneric methods.*

Description

Define rgeneric methods.

The rgeneric default method.

Usage

```
rgeneric(model, debug = FALSE, compile = TRUE, optimize = TRUE, ...)

## Default S3 method:
rgeneric(model, debug = FALSE, compile = TRUE, optimize = TRUE, ...)
```

Arguments

`model`	the model defined as a function. See the 'rgeneric' vignette from the INLA package.
`debug`	logical indicating debug state.
`compile`	logical indicating to compile the model.
`optimize`	logical indicating if only the elements of the precision matrix are returned.
`...`	additional arguments to be used internally for the model, for example, additional data.

Value

a `inla.rgeneric` object.

`rgeneric-class` *The `rgeneric` class for `rgeneric()`.*

Description

The `rgeneric` class for `rgeneric()`.

Usage

```
## S3 method for class 'inla.rgeneric'
graph(model, ...)

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

## S3 method for class 'inla.rgeneric'
initial(model)

## S3 method for class 'inla.rgeneric'
mu(model, theta)

## S3 method for class 'inla.rgeneric'
prior(model, theta)
```

Arguments

`model`	a `inla.rgeneric` model object
`...`	additional parameter such as 'theta' If 'theta' is not supplied, initial will be taken.
`theta`	the parameter.

Functions

- `graph(inla.rgeneric)`: The graph method for 'inla.rgeneric'
- `prec(inla.rgeneric)`: The precision method for an `inla.rgeneric` object.
- `initial(inla.rgeneric)`: The initial method for 'inla.rgeneric'
- `mu(inla.rgeneric)`: The mu method for 'inla.rgeneric'
- `prior(inla.rgeneric)`: The prior metho for 'inla.rgeneric'

`Sparse` *To store in i,j,x sparse matrix format*

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 Sparse.

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