Package 'graphpcor'

July 22, 2025

Type Package

Title Models for Correlation Matrices Based on Graphs

Version 0.1.12

Maintainer Elias Krainski <eliaskrainski@gmail.com>

Description Implement some models for
correlation/covariance matrices including two approaches
to model correlation matrices from a graphical structure.
One use latent parent variables as proposed in
Sterrantino et. al. (2024) <doi:10.48550/arXiv.2312.06289>.
The other uses a graph to specify conditional
relations between the variables.
The graphical structure makes correlation matrices
interpretable and avoids the quadratic increase of
parameters as a function of the dimension.
In the first approach a natural sequence of simpler
models along with a complexity penalization is used.
The second penalizes deviations from a base model.
These can be used as prior for model parameters,
considering C code through the 'cgeneric' interface
for the 'INLA' package (<https://www.r-inla.org>).
This allows one to use these models as building
blocks combined and to other latent Gaussian models
in order to build complex data models.

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, graph, numDeriv

Imports methods, stats, utils, Rgraphviz

Suggests INLA (>= 24.02.09)

BuildVignettes true

Author Elias Krainski [cre, aut, cph] (ORCID:
<https://orcid.org/0000-0002-7063-2615>),
Denis Rustand [aut, cph] (ORCID:
<https://orcid.org/0000-0001-9708-5220>),
Anna Freni-Sterrantino [aut, cph] (ORCID:
<https://orcid.org/0000-0002-6602-6209>),
Janet van Niekerk [aut, cph] (ORCID:
<https://orcid.org/0000-0002-4334-2057>),
Haavard Rue' [aut] (ORCID: <https://orcid.org/0000-0002-0222-1881>)

Repository CRAN

Date/Publication 2025-04-27 23:20:02 UTC

Contents

`cgeneric_generic0` *Build an* `inla.cgeneric` *to implement a model whose precision has a conditional precision parameter. See details. This uses the cgeneric interface that can be used as a model in a* `INLA` `f()` *model component.*

Description

Build an `inla.cgeneric` to implement a model whose precision has a conditional precision parameter. See details. This uses the cgeneric interface that can be used as a model in a `INLA f()` model component.

Usage

```
cgeneric_generic0(
  R,
  param,
  constr = TRUE,
  scale = TRUE,
  debug = FALSE,
  useINLAprecomp = TRUE,
  libpath = NULL
)

cgeneric_iid(
  n,
  param,
  constr = FALSE,
  scale = TRUE,
  debug = FALSE,
  useINLAprecomp = TRUE,
  libpath = NULL
)
```

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 mnodel. See detais.
`debug`	integer, default is zero, indicating the verbose level. Will be used as logical by INLA.
`useINLAprecomp`	logical, default is TRUE, indicating if it is to be used the shared object precompiled by INLA. This is not considered if 'libpath' is provided.
`libpath`	string, default is NULL, with the path to the shared object.
`n`	size of the model

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 `inla.cgeneric`, `cgeneric()` object.

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.

`cgeneric_graphpcor` *Build an* `inla.cgeneric` *for a graph, see* `graphpcor()`

Description

From either a `graph` (see `graph()`) or a square matrix (used as a graph), creates an `inla.cgeneric` (see `cgeneric()`) to implement the Penalized Complexity prior using the Kullback-Leibler divergence - KLD from a base graphpcor.

Usage

```
cgeneric_graphpcor(
  graph,
  lambda,
  base,
  sigma.prior.reference,
  sigma.prior.probability,
```

```
  params.id,
  low.params.fixed,
  debug = FALSE,
  useINLAprecomp = TRUE,
  libpath = NULL
)
```

Arguments

graph	a graphpcor (see graphpcor()) or a square matrix (to be used as a graph) to define the precision structure of the model.
lambda	the parameter for the exponential prior on the radius of the sphere, see details.
base	numeric vector with length m, m is the number of edges in the graph, or matrix with the reference correlation model against what the KLD will be evaluated. If it is a vector, a correlation matrix is defined considering the graph model and this vector as the parameters in the lower triangle matrix L. If it is a matrix, it will be checked if the graph model can generates this.
sigma.prior.reference	numeric vector with length n, n is the number of nodes (variables) in the graph, as the reference standard deviation to define the PC prior for each marginal variance parameters. If missing, the model will be assumed for a correlation. If a length n vector is given and sigma.prior.reference is missing, it will be used as known square root of the variances. NOTE: params.id will be applied here as sigma.prior.reference[params.id[1:n]].
sigma.prior.probability	numeric vector with length n to set the probability statement of the PC prior for each marginal variance parameters. The probability statement is P(sigma < sigma.prior.reference) = p. If missing, all the marginal variances are considered as known, as described in sigma.prior.reference. If a vector is given and a probability is NA, 0 or 1, the corresponding sigma.prior.reference will be used as fixed. NOTE: params.id will be applied here as sigma.prior.probability[params.id[1:n
params.id	integer ordered vector with length equals to n+m to specify common parameter values. If missing it is assumed 1:(n+m) and all parameters are assumed distinct. The first n indexes the square root of the marginal variances and the remaining indexes the edges parameters. Example: By setting params.id = c(1,1,2,3, 4,5,5,6), the first two standard deviations are common and the second and third edges parameters are common as well, giving 6 unknown parameters in the model.
low.params.fixed	numeric vector of length m providing the value(s) at which the lower parameter(s) of the L matrix to be fixed and not estimated. NA indicates not fixed and all are set to be estimated by default. Example: with low.params.fixed = c(NA, -1, NA, 1) the first and the third of these parameters will be estimated while the second is fixed and equal to -1 and the forth is fixed and equal to 1. NOTE: params.id will be applied here as low.params.fixed[params.id[(n+1:m)]-n+1], thus the provided examples give NA -1 -1 NA and so the second and third low L parameters are fixed to -1.

debug	integer, default is zero, indicating the verbose level. Will be used as logical by INLA.
useINLAprecomp	logical, default is TRUE, indicating if it is to be used the shared object precompiled by INLA. This is not considered if 'libpath' is provided.
libpath	string, default is NULL, with the path to the shared object.

Value

a inla.cgeneric, cgeneric() object.

`cgeneric_LKJ` *Build an* `inla.cgeneric` *object to implement the LKG prior for the correlation matrix.*

Description

Build an `inla.cgeneric` object to implement the LKG prior for the correlation matrix.

Usage

```
cgeneric_LKJ(n, eta, debug = FALSE, useINLAprecomp = TRUE, libpath = NULL)
```

Arguments

n	integer to define the size of the matrix
eta	numeric greater than 1, the parameter
debug	integer, default is zero, indicating the verbose level. Will be used as logical by INLA.
useINLAprecomp	logical, default is TRUE, indicating if it is to be used the shared object precompiled by INLA. This is not considered if 'libpath' is provided.
libpath	string, default is NULL, with the path to the shared object.

Details

The parametrization uses the hypershere decomposition, as proposed in Rapisarda, Brigo and Mercurio (2007). consider $\theta[k] \in [0, \infty], k = 1, ..., m = n(n-1)/2$ from $\theta[k] \in [0, \infty], k = 1, ..., m = n(n-1)/2$ compute $x[k] = pi/(1 + exp(-theta[k]))$ organize it as a lower triangle of a $n \times n$ matrix

$$B[i,j] = \begin{vmatrix} cos(x[i,j]), j = 1 \\ cos(x[i,j]) prod_{k=1}^{j-1} sin(x[i,k]), 2 <= j <= i-1 \\ prod_{k=1}^{j-1} sin(x[i,k]), j = i \\ 0, j+1 <= j <= n \end{vmatrix}$$

Result

$$\gamma[i,j] = -log(sin(x[i,j]))$$

$$KLD(R) = \sqrt{(}2 \sum_{i=2}^{n} \sum_{j=1}^{i-1} \gamma[i,j]$$

Value

a inla.cgeneric, cgeneric() object.

References

Rapisarda, Brigo and Mercurio (2007). Parameterizing correlations: a geometric interpretation. IMA Journal of Management Mathematics (2007) 18, 55-73. <doi 10.1093/imaman/dpl010>

`cgeneric_pc_correl` *Build an* `inla.cgeneric` *to implement the PC prior, proposed on Simpson et. al. (2007), for the correlation matrix parametrized from the hypershere decomposition, see details.*

Description

Build an `inla.cgeneric` to implement the PC prior, proposed on Simpson et. al. (2007), for the correlation matrix parametrized from the hypershere decomposition, see details.

Usage

```
cgeneric_pc_correl(
  n,
  lambda,
  debug = FALSE,
  useINLAprecomp = TRUE,
  libpath = NULL
)
```

Arguments

`n`	integer to define the size of the matrix
`lambda`	numeric (positive), the penalization rate parameter
`debug`	integer, default is zero, indicating the verbose level. Will be used as logical by INLA.
`useINLAprecomp`	logical, default is TRUE, indicating if it is to be used the shared object pre-compiled by INLA. This is not considered if 'libpath' is provided.
`libpath`	string, default is NULL, with the path to the shared object.

Details

The hypershere decomposition, as proposed in Rapisarda, Brigo and Mercurio (2007) consider $\theta[k] \in [0, \infty], k = 1, ..., m = n(n-1)/2$ compute $x[k] = pi/(1 + exp(-\theta[k]))$ organize it as a lower triangle of a $n \times n$ matrix

$$B[i,j] = \begin{cases} cos(x[i,j]) & j = 1 \\ cos(x[i,j]) prod_{k=1}^{j-1} sin(x[i,k]) & 2 <= j <= i-1 \\ prod_{k=1}^{j-1} sin(x[i,k]) & j = i \\ 0 & j+1 <= j <= n \end{cases}$$

Result

$$\gamma[i,j] = -log(sin(x[i,j]))$$

$$KLD(R) = \sqrt{(}2\sum_{i=2}^{n}\sum_{j=1}^{i-1}\gamma[i,j]$$

Value

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

References

Daniel Simpson, H\aa vard Rue, Andrea Riebler, Thiago G. Martins and Sigrunn H. S\o rbye (2017). Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors Statistical Science 2017, Vol. 32, No. 1, 1–28. <doi 10.1214/16-STS576>

Rapisarda, Brigo and Mercurio (2007). Parameterizing correlations: a geometric interpretation. IMA Journal of Management Mathematics (2007) 18, 55-73. <doi 10.1093/imaman/dpl010>

`cgeneric_pc_prec_correl`

Build an `inla.cgeneric` *to implement the PC-prior of a precision matrix as inverse of a correlation matrix.*

Description

Build an `inla.cgeneric` to implement the PC-prior of a precision matrix as inverse of a correlation matrix.

Usage

```
cgeneric_pc_prec_correl(
  n,
  lambda,
  theta.base,
  debug = FALSE,
  useINLAprecomp = TRUE,
  libpath = NULL
)
```

Arguments

`n`	integer to define the size of the matrix
`lambda`	numeric (positive), the penalization rate parameter
`theta.base`	numeric vector with the model parameters at the base model
`debug`	integer, default is zero, indicating the verbose level. Will be used as logical by INLA.

`useINLAprecomp` logical, default is TRUE, indicating if it is to be used the shared object pre-compiled by INLA. This is not considered if 'libpath' is provided.

`libpath` string, default is NULL, with the path to the shared object.

Details

The precision matrix parametrization step 1:

$$Q0 = \begin{bmatrix} 1 & & & \\ \theta_1 & 1 & & \\ \theta_2 & \theta_n & & \\ \vdots & & \dots & \ddots \\ \theta_{n-1} & \theta_{2n-3} \dots & \theta_m & 1 \end{bmatrix}$$

step 2: $V = Q0^{-1}$

step 3: $S = diag(V)^{1/2}$

step 4: $C = SVS$

step 5: $Q = C^{-1}$

$$p(Q|\lambda) = p(\theta[1:m]|lambda) =$$

$$p_C(C(Q))|JacobianC(Q)|$$

where p_C is the PC-prior for correlation, see section 6.2 of Simpson et. al. (2017), which is based on the hypersphere decomposition.

The hypershere decomposition, as proposed in Rapisarda, Brigo and Mercurio (2007) consider $\theta[k] \in [0, \infty], k = 1, ..., m = n(n-1)/2$ compute $x[k] = pi/(1 + exp(-theta[k]))$ organize it as a lower triangle of a $n \times n$ matrix

$$B[i,j] = \begin{cases} cos(x[i,j]) & j = 1 \\ cos(x[i,j]) prod_{k=1}^{j-1} sin(x[i,k]) & 2 <= j <= i-1 \\ prod_{k=1}^{j-1} sin(x[i,k]) & j = i \\ 0 & j+1 <= j <= n \end{cases}$$

Result

$$\gamma[i,j] = -log(sin(x[i,j]))$$

$$KLD(R) = \sqrt{(}2 \sum_{i=2}^{n} \sum_{j=1}^{i-1} \gamma[i,j]$$

Value

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

References

Daniel Simpson, H\aa vard Rue, Andrea Riebler, Thiago G. Martins and Sigrunn H. S\o rbye (2017). Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors Statistical Science 2017, Vol. 32, No. 1, 1–28. <doi 10.1214/16-STS576>

Rapisarda, Brigo and Mercurio (2007). Parameterizing correlations: a geometric interpretation. IMA Journal of Management Mathematics (2007) 18, 55-73. <doi 10.1093/imaman/dpl010>

`cgeneric_treepcor` *Build an* `cgeneric` *for* `treepcor()`

Description

This set the necessary data to implement the penalized complexity prior for a correlation matrix considering a three as proposed in Sterrantino et. al. 2025

Usage

```
cgeneric_treepcor(
  graph,
  lambda,
  sigma.prior.reference,
  sigma.prior.probability,
  debug = FALSE,
  useINLAprecomp = TRUE,
  libpath = NULL
)
```

Arguments

`graph`	object of class `treepcor` for the model specification.
`lambda`	the lambda parameter for the graph correlation prior.
`sigma.prior.reference`	a vector with the reference values to define the prior for the standard deviation parameters.
`sigma.prior.probability`	a vector with the probability values to define the prior for the standard deviation parameters.
`debug`	integer, default is zero, indicating the verbose level. Will be used as logical by INLA.
`useINLAprecomp`	logical, default is TRUE, indicating if it is to be used the shared object pre-compiled by INLA. This is not considered if 'libpath' is provided.
`libpath`	string, default is NULL, with the path to the shared object.

Details

The correlation prior as in the paper depends on the lambda value. The prior for each $sigma_i$ is the Penalized-complexity prior which can be defined from the following probability statement P(sigma > U) = a. where "U" is a reference value and "a" is a probability. The values "U" and probabilities "a" for each $sigma_i$ are passed in the `sigma.prior.reference` and `sigma.prior.probability` arguments. If a=0 then U is taken to be the fixed value of the corresponding sigma. E.g. if there are three sigmas in the model and one supply sigma.prior.reference = c(1, 2, 3) and sigma.prior.probability = c(0.05, 0.0, 0.01) then the sigma is fixed to 2 and not estimated.

Value

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

See Also

`treepcor()` and `cgeneric()`

---

cgeneric_Wishart *Build an* `inla.cgeneric` *to implement the Wishart prior for a precision matrix.*

Description

Build an `inla.cgeneric` to implement the Wishart prior for a precision matrix.

Usage

```
cgeneric_Wishart(
  n,
  dof,
  R,
  debug = FALSE,
  useINLAprecomp = TRUE,
  libpath = NULL
)
```

Arguments

`n`	the size of the precision matrix
`dof`	degrees of freedom model parameter
`R`	lower triangle of the scale matrix parameter
`debug`	integer, default is zero, indicating the verbose level. Will be used as logical by INLA.
`useINLAprecomp`	logical, default is TRUE, indicating if it is to be used the shared object pre-compiled by INLA. This is not considered if 'libpath' is provided.
`libpath`	string, default is NULL, with the path to the shared object.

Details

For a random $p \times p$ precision matrix Q, given the parameters d and R, respectively scalar degree of freedom and the *inverse* scale $p \times p$ matrix the Wishart density is

$$|Q|^{(d-p-1)/2} \mathrm{e}^{-tr(RQ)/2} |R|^{p/2} 2^{-dp/2} \Gamma_p(n/2)^{-1}$$

Value

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

---

dLKJ *The LKJ density for a correlation matrix*

Description

The LKJ density for a correlation matrix

Usage

```
dLKJ(R, eta, log = FALSE)
```

Arguments

`R`	correlation matrix
`eta`	numeric, the prior parameter
`log`	logical indicating if the log of the density is to be returned, default = FALSE

Value

numeric as the (log) density

---

graphpcor *The* `graphpcor` *generic method for graphpcor*

Description

The `graphpcor` generic method for graphpcor

Usage

```
graphpcor(...)
```

Arguments

`...`	either a list of formulae or a matrix

Value

a `graphpcor` object

graphpcor-class *Set a graph whose nodes and edges represent variables and conditional distributions, respectively.*

Description

Set a graph whose nodes and edges represent variables and conditional distributions, respectively.

Usage

```
## S3 method for class 'formula'
graphpcor(...)

## S3 method for class 'matrix'
graphpcor(...)

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

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

## S3 method for class 'graphpcor'
dim(x, ...)

## S4 method for signature 'graphpcor'
edges(object, which, ...)

## S4 method for signature 'graphpcor,ANY'
plot(x, y, ...)

## S3 method for class 'graphpcor'
Laplacian(graph)

## S4 method for signature 'graphpcor'
chol(x, ...)

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

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

Arguments

...	list of formula used to define the edges.
x	a graphpcor object

object	graphpcor object
which	not used
y	graphpcor
graph	graphpcor object, see graphpcor.
model	graphpcor model object

Details

The terms in the formula do represent the nodes. The ~ is taken as link.

Methods (by generic)

- edges(graphpcor): Extract the edges of a graphcor to be used for plot
- plot(x = graphpcor, y = ANY): The plot method for graphpcor
- chol(graphpcor): Build the unite diagonal lower triangle matrix
- vcov(graphpcor): The vcov method for a graphpcor

Functions

- graphpcor(formula): A graphpcor is a graph where a node represents a variable and an edge a conditional distribution.
- graphpcor(matrix): Build a graphpcor from a matrix
- print(graphpcor): The print method for graphpcor
- summary(graphpcor): The summary method for graphpcor
- dim(graphpcor): The dim method for graphpcor
- Laplacian(graphpcor): The Laplacian method for a graphpcor
- prec(graphpcor): The precision method for 'graphpcor'

Examples

```
g1 <- graphpcor(x ~ y, y ~ v, v ~ z, z ~ x)
g1
summary(g1)
plot(g1)
prec(g1)
```

`hessian.graphpcor` *Evaluate the hessian of the KLD for a* `graphpcor` *correlation model around a base model.*

Description

Evaluate the hessian of the KLD for a `graphpcor` correlation model around a base model.

Usage

```
## S3 method for class 'graphpcor'
hessian(func, x, method = "Richardson", method.args = list(), ...)
```

Arguments

`func`	model definition of a graphical model. This can be either a matrix or a 'graphpcor'.
`x`	either a reference correlation matrix or a numeric vector with the parameters for the reference 'graphpcor' model.
`method`	see `numDeriv::hessian()`
`method.args`	see `numDeriv::hessian()`
`...`	use to pass the decomposition method, as a character to specify which one is to be used to compute H^0.5 and H^(1/2).

Value

list containing the hessian, its 'square root', inverse 'square root' along with the decomposition used

Examples

```
g <- graphpcor(x1 ~ x2 + x3, x2 ~ x4, x3 ~ x4)
ne <- dim(g)
gH0 <- hessian(g, rep(-1, ne[2]))
## alternatively
C0 <- vcov(g, theta = rep(c(0,-1), ne))
all.equal(hessian(g, C0), gH0)
```

`inla.cgeneric-class` `inla.cgeneric` *class, short* `cgeneric`*, to define a* `INLA::cgeneric()` *latent model*

Description

This organize data needed on the C interface for building latent models, which are characterized from a 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, ...)

## Default S3 method:
cgeneric(model, debug = FALSE, useINLAprecomp = TRUE, libpath = NULL, ...)

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

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

initial(model)
```

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

mu(model, theta)

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

prior(model, theta)

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

graph(model, ...)

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

Q(model, ...)

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

## S3 method for class 'graphpcor'
cgeneric(...)

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

## S3 method for class 'treepcor'
cgeneric(...)
```

Arguments

model	an object inla.cgeneric object.
...	arguments passed on.
debug	integer, default is zero, indicating the verbose level. Will be used as logical by INLA.
useINLAprecomp	logical, default is TRUE, indicating if it is to be used the shared object pre-compiled by INLA. This is not considered if 'libpath' is provided.
libpath	string, default is NULL, with the path to the shared 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. If missing or FALSE, the graph and precision are as a sparse matrix. If TRUE, graph only return the row/col indexes and precision return only the elements as a vector.

X	inla.cgeneric or inla.rgeneric
Y	inla.cgeneric or inla.rgeneric
FUN	see kronecker
make.dimnames	see kronecker

Value

a inla.cgeneric, cgeneric() object.

depends on cmd

Methods (by class)

- cgeneric(default): This calls INLA::inla.cgeneric.define()
- cgeneric(character): Method for when model is a character. E.g. cgeneric(model = "generic0") calls cgeneric_generic0
- cgeneric(graphpcor): The cgeneric method for graphpcor uses cgeneric_graphpcor()
- cgeneric(treepcor): The cgeneric method for treepcor, uses cgeneric_treepcor()

Functions

- cgeneric_get(): cgeneric_get is an internal function used by graph, prec, initial, mu or prior methods for inla.cgeneric
- initial(): Retrieve the initial model parameter(s).
- initial(inla.cgeneric): Retrive the initial parameter(s) of an inla.cgeneric model.
- mu(): Evaluate the mean.
- mu(inla.cgeneric): Evaluate the mean for an inla.cgeneric model.
- prior(): Evaluate the log-prior.
- prior(inla.cgeneric): Evaluate the prior for an inla.cgeneric model
- graph(): Retrieve the graph
- graph(inla.cgeneric): Retrieve the graph of an inla.cgeneric object
- Q(): Evaluate prec() on a model
- prec(inla.cgeneric): Evaluate prec() on an inla.cgeneric object
- kronecker(X = inla.cgeneric, Y = inla.cgeneric): Kronecker (product) between two inla.cgeneric models as a method for kronecker()

See Also

INLA::cgeneric()

inla.rgeneric-class inla.rgeneric *class, short* rgeneric*, to define a* INLA::rgeneric() *latent model*

Description

inla.rgeneric class, short rgeneric, to define a INLA::rgeneric() latent model

Define rgeneric methods.

The rgeneric default method.

Usage

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

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

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

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

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

## 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

X	inla.cgeneric or inla.rgeneric
Y	inla.cgeneric or inla.rgeneric
FUN	see kronecker

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

Value

a inla.rgeneric object.

Functions

- kronecker(X = inla.cgeneric, Y = inla.rgeneric): Kronecker (product) between a inla.cgeneric model and a inla.rgeneric model as a method for kronecker()
- kronecker(X = inla.rgeneric, Y = inla.cgeneric): Kronecker (product) between a inla.rgeneric model and a inla.cgeneric model as a method for kronecker()
- kronecker(X = inla.rgeneric, Y = inla.rgeneric): Kronecker (product) between a inla.rgeneric model and a inla.rgeneric model as a method for kronecker()
- 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'

---

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

Laplacian *The Laplacian of a graph*

Description

The (symmetric) Laplacian of a graph is a square matrix with dimention equal the number of nodes. It is defined as

$$L_{ij} = n_i \text{ if } i = j, -1 \text{ if } i \sim j, 0 \text{ otherwise}$$

where i~j means that there is an edge between nodes i and j and n_i is the number of edges including node i.

Usage

```
Laplacian(graph)

## Default S3 method:
Laplacian(graph)

## S3 method for class 'matrix'
Laplacian(graph)
```

Arguments

graph	an object that inherits a matrix class

Value

matrix as the Laplacian of a graph

Methods (by class)

- Laplacian(default): The Laplacian default method (none)
- Laplacian(matrix): The Laplacian of a matrix

Lprec *Precision matrix parametrization helper functions.*

Description

Precision matrix parametrization helper functions.

Usage

```
Lprec(theta, p, ilowerL)

fillLprec(L, lfi)

theta2Lprec2C(theta, p, ilowerL)
```

Arguments

theta	numeric vector of length m.
p	numeric giving the dimention of Q. If missing, p = (1+sqrt(1+8*length(theta))) and Q is assumed to be dense.
ilowerL	numeric vector as index to (lower) L to be filled with theta. Default is missing and Q is assumed to be dense.
L	matrix as the lower triangle containing the Cholesky decomposition of a precision matrix
lfi	indicator of fill-in elements

Details

The precision matrix definition consider m parameters for the lower part of L. If Q is dense, then m = p(p-1)/2, else m = length(ilowerL). Then the precision is defined as $Q(\theta) = L(\theta)L(\theta)^T$

Value

matrix as the Cholesky factor of a precision matrix as the inverse of a correlation

a matrix whose elements at the lower triangle are the filled in elements of the Cholesky decomposition of a precision matrix and diagonal elements as 1:p.

Functions

- fillLprec(): Function to fill-in a Cholesky matrix
- theta2Lprec2C(): Internal function to build C

Examples

```
theta1 <- c(1, -1, -2)
Lprec(theta1)
theta2 <- c(0.5, -0.5, -1, -1)
Lprec(theta2, 4, c(2,4,7,12))
```

---

prec *The* prec *method*

Description

The prec method

Usage

```
prec(model, ...)

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

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

Arguments

model	a model object
...	additional arguments

Value

a precision matrix

Functions

- prec(default): The default precision method computes the inverse of the variance
- prec(inla): Define the prec method for an inla output object

rphi2x *Functions for the mapping between spherical and Euclidean coordinates.*

Description

Functions for the mapping between spherical and Euclidean coordinates.

Usage

```
rphi2x(rphi)

x2rphi(x)

rtheta(n, lambda = 1, R, theta.base)

dtheta(theta, lambda, theta.base, H.elements)

KLD10(C1, C0)

theta2H(theta)
```

Arguments

rphi	numeric vector where the first element is the radius and the remaining are the angles
x	parameters in the Euclidian space to be converted
n	integer to define the size of the correlation matrix
lambda	numeric as the parameter for the Exponential distribution of the radius
R	scaling matrix (square root of the Hessian around the base model)
theta.base	numeric vector of the base model
theta	numeric vector of length m.
H.elements	list output of theta2H
C1	is a correlation matrix.
C0	is a correlation matrix of the base model.

Details

see N-sphere/Euclidian

compute C1 using 'theta2C' on theta with

$$KLD = 0.5(tr(C0^{-1}C1) - p + ... - log(|C1|) + log(|C0|))$$

Functions

- `x2rphi()`: Tranform from Euclidian coordinates to spherical
- `rtheta()`: Drawn samples from the PC-prior for correlation
- `dtheta()`: PC-prior density for the correlation matrix
- `KLD10()`: Compute the KLD with respect to a base model
- `theta2H()`: Compute the hessian, its svd and some elements

`theta2correl` *Build the correlation matrix parametrized from the hypershere decomposition, see details.*

Description

Build the correlation matrix parametrized from the hypershere decomposition, see details.

Usage

```
theta2correl(theta, fromR = TRUE)

theta2gamma2L(theta, fromR = TRUE)

rcorrel(p, lambda)
```

Arguments

`theta`	numeric vector with length equal n(n-1)/2
`fromR`	logical indicating if theta is in R. If FALSE, assumes $\theta[k] \in (0, pi)$.
`p`	integer to specify the matrix dimension
`lambda`	numeric as the penalization parameter. If missing it will be assumed equal to zero. The lambda=0 case means no penalization and a random correlation matrix will be drawn. Please see section 6.2 of the PC-prior paper, Simpson et. al. (2017), for details.

Details

The hypershere decomposition, as proposed in Rapisarda, Brigo and Mercurio (2007) consider $\theta[k] \in [0, \infty], k = 1, ..., m = n(n-1)/2$ compute $x[k] = pi/(1 + exp(-theta[k]))$ organize it as a lower triangle of a $n \times n$ matrix

$$|cos(x[i,j]), j = 1$$

$$B[i,j] = |cos(x[i,j]) prod_{k=1}^{j-1} sin(x[i,k]), 2 <= j <= i-1$$

$$|prod_{k=1}^{j-1} sin(x[i,k]), j = i$$

$$|0, j+1 <= j <= n$$

Result

$$\gamma[i,j] = -log(sin(x[i,j]))$$

$$KLD(R) = \sqrt{(}2\sum_{i=2}^{n}\sum_{j=1}^{i-1}\gamma[i,j]$$

Value

a correlation matrix

Lower triangular n x n matrix

Functions

- `theta2gamma2L()`: Build a lower triangular matrix from a parameter vector. See details.
- `rcorrel()`: Drawn a random sample correlation matrix

References

Rapisarda, Brigo and Mercurio (2007). Parameterizing correlations: a geometric interpretation. IMA Journal of Management Mathematics (2007) 18, 55-73. <doi 10.1093/imaman/dpl010>

Simspon et. al. (2017). Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors. Statist. Sci. 32(1): 1-28 (February 2017). <doi: 10.1214/16-STS576>

`treepcor` *Define a tree used to model correlation matrices using a shared latent variables method represented by a tree, whose nodes represent the two kind of variables: children and parent. See treepcor.*

Description

Define a tree used to model correlation matrices using a shared latent variables method represented by a tree, whose nodes represent the two kind of variables: children and parent. See treepcor.

Usage

```
treepcor(...)
```

Arguments

...	a list of formula used as relationship to define a three for correlation modeling, see `treepcor()`. Parent nodes shall be in the right side while children (or parent with a parent) in the left side.

Details

The children variables are those with an ancestor (parent), and are identified as c1, ..., cn, where n is the total number of children variables. The variables are identified as p1, ..., pm, where the m is the number of parent variables. The main parent (fist) should be identified as p1. Parent variables (except p1) have an ancestor, which is a parent variable.

Value

a treepcor object

Examples

```
g1 <- treepcor(p1 ~ c1 + c2 - c3)
g1
summary(g1)
plot(g1)
prec(g1)
prec(g1, theta = 0)

g2 <- treepcor(p1 ~ c1 + c2 + p2,
          p2 ~ c3 - c4)
g2
summary(g2)
plot(g2)
prec(g2)
prec(g2, theta = c(0, 0))

g3 <- treepcor(p1 ~ -p2 + c1 + c2,
          p2 ~ c3)
g3
summary(g3)
plot(g3)
prec(g3)
prec(g3, theta = c(0,0))
```

---

treepcor-class *Set a tree whose nodes represent the two kind of variables: children and parent.*

Description

Set a tree whose nodes represent the two kind of variables: children and parent.

Usage

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

## S3 method for class 'treepcor'
```

```
summary(object, ...)

## S3 method for class 'treepcor'
dim(x, ...)

## S4 method for signature 'treepcor'
drop1(object)

## S4 method for signature 'treepcor'
edges(object, which, ...)

## S4 method for signature 'treepcor,ANY'
plot(x, y, ...)

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

etreepcor2precision(d.el)

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

etreepcor2variance(d.el)
```

Arguments

x	treepcor object
...	usde to pass theta as a numeric vector with the model parameters
object	treepcor
which	not used (TO DO:)
y	not used
model	treepcor
d.el	list of the first n edges of a treepcor.

Methods (by generic)

- drop1(treepcor): The drop1 method for a treepcor
- edges(treepcor): Extract the edges of a treepcor to be used for plot
- plot(x = treepcor, y = ANY): The plot method for a treepcor
- vcov(treepcor): The vcov method for a treepcor

Functions

- print(treepcor): The print method for a treepcor
- summary(treepcor): The summary method for a treepcor
- dim(treepcor): The dim for a treepcor

- `prec(treepcor)`: The `prec` for a `treepcor`
- `etreepcor2precision()`: Internal function to extract elements to build the precision from the `treepcor` edges.
- `etreepcor2variance()`: Internal function to extract elements to build the covariance matrix from a `treepcor`.

Index