Package 'graphpcor'

January 15, 2026

Type Package

Title Models for Correlation Matrices Based on Graphs

Version 0.1.15

Maintainer Elias Teixeira 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:10.1007/s10260-025-00788-y>.
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 7.3.3

NeedsCompilation yes

Depends R (>= 4.3), Matrix, INLAtools, graph, numDeriv

Imports methods, stats, utils, Rgraphviz

Suggests knitr, INLA (>= 25.12.16)

BuildVignettes true

VignetteBuilder knitr

Author Elias Teixeira 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 2026-01-15 08:40:01 UTC

Contents

`basecor-class` *Information for a base model for correlation matrices*

Description

The `basecor` class contain a correlation matrix `base`, the parameter vector `theta`, that generates or is generated by `base`, the dimention `p`, the index `itheta` for `theta` in the (lower) Cholesky, and the Hessian around it `I0`, see details.

Usage

```
basecor(base, p, parametrization = "cpc", itheta)

## S3 method for class 'numeric'
basecor(base, p, parametrization = "cpc", itheta)

## S3 method for class 'matrix'
basecor(base, p, parametrization = "cpc", itheta)

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

Arguments

base	numeric vector/matrix used to define the base correlation matrix. If numeric vector with length 'm', 'm' should be 'p(p-1)/2' in the dense model case and 'length(itheta)' in the sparse model case.
p	integer with the dimension, the number of rows/columns of the correlation matrix.
parametrization	character to specify the parametrization used. The available ones are "cpc" (or "CPC") or "sap" (or "SAP"). See Details. The default is "cpc".
itheta	integer vector to specify the (vectorized) position where 'theta' will be placed in the (initial, before fill-in) Cholesky (lower triangle) factor. Default is missing and assumes the dense case for when `itheta = which(lower.tri(...))`.
x	a basecor object.
...	further arguments passed on.

Details

For 'parametrization' = "CPC" or 'parametrization' = "cpc": The Canonical Partial Correlation - CPC parametrization, Lewandowski, Kurowicka, and Joe (2009), compute $r[k]$ = tanh($\theta[k]$), for $k = 1, ..., m$, and the two $p \times p$ matrices

$$A = \begin{bmatrix} 1 & & & & \\ r_1 & 1 & & & \\ r_2 & r_p & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ r_{p-1} & r_{2p-3} & \dots & r_m & 1 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & & & & \\ \sqrt{1-r_1^2} & 1 & & & \\ \sqrt{1-r_2^2} & \sqrt{1-r_p^2} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ \sqrt{1-r_{p-1}^2} & \sqrt{r_{2p-3}^2} & \dots & \sqrt{1-r_m^2} & 1 \end{bmatrix}$$

The matrices A and B are then used to build the Cholesky factor of the correlation matrix, given as

$$L = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ A_{2,1} & B_{2,1} & 0 & \dots & 0 \\ A_{3,1} & A_{3,2}B_{3,1} & B_{3,1}B_{3,2} & & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ A_{p,1} & A_{p,2}B_{p,1} & \dots & A_{p,p-1}\prod_{k=1}^{p-1} B_{p,k} & \prod_{k=1}^{p-1} B_{p,k} \end{bmatrix}$$

Note: The determinant of the correlation matriz is

$$\prod_{i=2}^{p}\prod_{j=1}^{i-1} B_{i,j} = \prod_{i=2}^{p} L_{i,i}$$

For 'parametrization' = "SAP" or 'parametrization' = "sap": The Standard Angles Parametrization - SAP, as described in Rapisarda, Brigo and Mercurio (2007), compute $x[k] = \pi/(1+\exp(-\theta[k]))$, for $k = 1, ..., m$, and the two $p \times p$ matrices

$$A = \begin{bmatrix} 1 & & & & \\ \cos(x_1) & 1 & & & \\ \cos(x_2) & \cos(x_p) & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ \cos(x_{p-1}) & \cos(x_{2p-3}) & \dots & \cos(x_m) & 1 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & & & & \\ \sin(x_1) & 1 & & & \\ \sin(x_2) & \sin(x_p) & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ \sin(x_{p-1}) & \sin(x_{2p-3}) & \dots & \sin(x_m) & 1 \end{bmatrix}$$

The decomposition of the Hessian matrix around the base model, `I0`, formally $\mathbf{I}(\theta_0)$, is numerically computed. This element has the following attributes: 'h.5' as $\mathbf{I}^{1/2}(\theta_0)$, and 'hneg.5' as $\mathbf{I}^{-1/2}(\theta_0)$.

Value

a `basecor` object

Functions

- `basecor()`: Build a `basecor` object.
- `basecor(numeric)`: Build a `basecor` from the parameter vector.
- `basecor(matrix)`: Build a `basecor` from a correlation matrix.
- `print(basecor)`: Print method for 'basecor'

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>

Lewandowski, Kurowicka and Joe (2009) Generating Random Correlation Matrices Based on Vines and Extended Onion Method. Journal of Multivariate Analysis 100: 1989–2001. <doi: 10.1016/j.jmva.2009.04.008>

Simpson, 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>

Examples

```
library(graphpcor)

## A correlation matrix
c0 <- matrix(c(1,.8,-.625, 0.8,1,-.5, -0.625,-.5,1), 3)

## build the 'basecor'
pc.c0 <- basecor(c0) ## base as matrix
```

```
pc.c0

## elements
pc.c0$base
pc.c0$theta
pc.c0$I0

## from 'theta'
th0 <- pc.c0$theta
pc.th0 <- basecor(th0) ## base as vector
pc.th0

## numerically the same
all.equal(c0, pc.th0$base)

## from a numeric vector (theta)
th2 <- c(-1, -0.5)
b2 <- basecor(th2, p = 3, itheta = c(2,3))
b2

## from the correlation matrix
b2c <- basecor(b2$base, itheta = c(2,3))

all.equal(th2, b2c$theta, tol = 1e-4)

## Hessian around the base (and its decomposition, etc.)
b2$I0
```

`basecor-utils` *Internal functions used by basecor*

Description

Internal functions used by basecor

Usage

```
cholcor(theta, p, itheta, parametrization = "cpc")

Hcorrel(theta, p, parametrization, itheta, C0, decomposition, ...)
```

Arguments

`theta`	numeric parameter vector.
`p`	integer with the dimension, the number of rows/columns of the correlation matrix.
`itheta`	integer vector to specify the (vectorized) position where 'theta' will be placed in the (initial, before fill-in) Cholesky (lower triangle) factor. Default is missing and assumes the dense case for when `itheta = which(lower.tri(...))`.

`parametrization`
: character to specify the parametrization used. The available ones are "cpc" (or "CPC") or "sap" (or "SAP"). See Details. The default is "cpc".

`C0`
: base correlation matrix.

`decomposition`
: character to inform which decomposition is to be applied to the hessian. The options are "eigen", "svd" and "chol".

`...`
: further arguments passed on.

Value

matrix with lower triangle as the Cholesky factor of a correlation matrix if parametrization is "cpc" or "sap" and of a precision matrix (of a correlation matrix) if parametrization is 'itp' (with 'd0' as the diagonal elements).

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

Functions

- `cholcor()`: Cholesky parametrization for a correlation matrix
- `Hcorrel()`: Evaluate the hessian of the KLD for a correlation model around a base model.

`basepcor-class` *Information for a base model for correlation matrices*

Description

The `basepcor` class contain a correlation matrix `base`, the parameter vector `theta`, that generates or is generated by `base`, the dimension `p`, the index `itheta` for `theta` in the (lower) Cholesky, and the Hessian around it `I0`, see details.

Usage

```
basepcor(base, p, itheta, d0)

## S3 method for class 'numeric'
basepcor(base, p, itheta, d0)

## S3 method for class 'matrix'
basepcor(base, p, itheta, d0)

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

Arguments

base	matrix (or numeric vector) as the base correlation (or parameter at the base model).
p	integer (needed if base is vector): the dimention.
itheta	integer vector or 'graphpcor' to specify the (vectorized) position where 'theta' is placed in the initial (before the fill-in) Cholesky (lower triangle) factor. If missing, default, assumes the dense case as itheta = which(lower.tri(...)), giving length(theta)=p(p-1)/2.
d0	numeric vector to specify the diagonal of the Cholesky factor for the initial precision matrix Q0. Default, if not provided, is d0 = p:1.
x	a basepcor object.
...	further arguments passed on.

Details

The Inverse Transform Parametrization - ITP, is applied by starting with a

$$\mathbf{L}^{(0)} = \begin{bmatrix} \text{p} & 0 & 0 & \dots & 0 \\ \theta_1 & \text{p-1} & 0 & \dots & 0 \\ \theta_2 & \theta_p & \text{p-2} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ \theta_{p-1} & \theta_{2p-3} & \dots & \theta_m & 1 \end{bmatrix}.$$

Then compute $\mathbf{Q}^{(0)} = \mathbf{L}\mathbf{L}^T$, $\mathbf{V}^{(0)} = (\mathbf{Q}^{(0)})^{-1}$ and $s_i^{(0)} = \sqrt{\mathbf{V}_{i,i}^{(0)}}$, and define $\mathbf{S}^{(0)} = \text{diag}(s_1^{(0)}, \dots, s_p^{(0)})$ in order to have $\mathbf{C} = \mathbf{S}^{-1}\mathbf{V}^{(0)}\mathbf{S}^{-1}$.

The decomposition of the Hessian matrix around the base model, I0, formally $\mathbf{I}(\theta_0)$, is numerically computed. This element has the following attributes: 'h.5' as $\mathbf{I}^{1/2}(\theta_0)$, and 'hneg.5' as $\mathbf{I}^{-1/2}(\theta_0)$.

Value

a basepcor object

a basepcor object

Functions

- basepcor(): Build a basepcor object.
- basepcor(numeric): Build a basepcor from the parameter vector.
- basepcor(matrix): Build a basepcor from a correlation matrix.
- print(basepcor): Print method for 'basepcor'

Examples

```
## p = 3, m = 2
bc <- basepcor(c(-1,-1), p = 3, itheta = c(2,3))
bc

round(solve(bc$base), 4)

all.equal(bc, basepcor(bc$base, itheta =c(2,3)))

## p = 4, m = 4
th2 <- c(0.5,-1,0.5,-0.3)
ith2 <- c(2,3,8,12)
b2 <- basepcor(th2, p = 4, itheta = ith2)
b2

Sparse(solve(b2$base), zeros.rm = TRUE)

all.equal(th2, basepcor(b2$base, itheta = ith2)$theta)

## Hessian around the base (and its decomposition, etc.)
b2$I0
```

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

Description

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

Usage

```
cgeneric_graphpcor(
  model,
  lambda,
  base,
  sigma.prior.reference,
  sigma.prior.probability,
  params.id,
  cor.params.fixed,
  ...
)
```

Arguments

`model`	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.probability` 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.

`cor.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 `cor.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 `cor.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`.

`...` additional arguments that will be passed on to `INLAtools::cgenericBuilder()`.

Value

`cgeneric` object.

See Also

`graphpcor()`

cgeneric_LKJ *Build an* cgeneric *model for the LKG prior on correlation matrix.*

Description

Build an cgeneric model for the LKG prior on correlation matrix.

Usage

```
cgeneric_LKJ(
  n,
  eta,
  sigma.prior.reference = rep(1, n),
  sigma.prior.probability = rep(NA, n),
  ...
)
```

Arguments

n	integer to define the size of the matrix
eta	numeric greater than 1, the parameter
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.
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.
...	additional arguments passed to INLAtools::cgeneric().

Details

It uses the Cannonical Partial Correlation (CPC), see basepcor() for details.

Value

a cgeneric object, see INLAtools::cgeneric() for details.

See Also

dLKJ() and basepcor()

Examples

```
## a cgeneric model for the LKJ prior
## for theta from the CPC parametrization
cglkj <- cgeneric(
    model = "LKJ", n = 3, eta = 2,
    useINLAprecomp = FALSE)##!is.na(packageCheck("INLA", "25.12.16")))

## correlation matrix, p = 3
cc <- matrix(c(1,.8,-.625, 0.8,1,-.5, -0.625,-.5,1), 3)

## CPC parametrization: C(theta)
(bb <- basecor(cc))
th <- bb$theta

## p(theta | eta)
prior(cglkj, theta = th)

## precision inverse
solve(prec(cglkj, theta = th))
solve(prec(cglkj, theta = c(0,0,0)))
solve(prec(cglkj, theta = c(1,1,1)))
```

`cgeneric_pc_correl` *Build an* `cgeneric` *object to implement the PC prior, proposed on Simpson et. al. (2007), as an informative prior, see details in* `basecor()`.

Description

Build an `cgeneric` object to implement the PC prior, proposed on Simpson et. al. (2007), as an informative prior, see details in `basecor()`.

Usage

```
cgeneric_pc_correl(
  n,
  lambda,
  base,
  sigma.prior.reference,
  sigma.prior.probability,
  params.id,
  cor.params.fixed,
  ...
)
```

Arguments

n	integer to define the size of the matrix
lambda	numeric (positive), the penalization rate parameter
base	matrix with base correlation matrix, or numeric vector representing the parameters of a base correlation matrix. See basepcor() for details.
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.probability 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.
cor.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 cor.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 cor.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.
...	additional arguments passed on to INLAtools::cgeneric().

Value

a cgeneric object, see INLAtools::cgeneric() for details.

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>

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

Description

Build an `cgeneric` object 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,
  shlib = 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 'shlib' is provided.
`shlib`	string, default is NULL, with the path to the shared object.

Details

The Canonical Partial Correlation - CPC parametrization, Lewandowski, Kurowicka, and Joe (2009). step 1: q_i = tanh(θ_i) step 2:

$$z = \begin{bmatrix} 1 & & & \\ q_1 & 1 & & \\ q_2 & q_n & & \\ \vdots & & \dots & \ddots \\ q_{n-1} & q_{2n-3} \cdots & q_m & 1 \end{bmatrix}$$

step 3: compute L such that the correlation matrix is

$$C = LL'$$

, a $n \times n$ (lower triangle) matrix

$$L_{i,j} = \begin{cases} 0 & i > j \\ 1 & i = j = 1 \\ z_{i,j} & j = 1 \\ prod_{k=1}^{j-1}\sqrt{(1 - z_{k,j^2})} & 1 < i = j \\ z_{i,j} prod_{k=1}^{j-1}\sqrt{(1 - z_{k,j}^2)} & 1 < j < i \end{cases}$$

The prior of the correlation matrix is given as

$$p(C) = |J_m| * l * exp(-l * r)/(2 * pi^(m - 1)$$

following a bijective transformation from

$$\theta[1:m] \in R^m to\{r, \phi[1:(m-1)]\}$$

where $\phi[1:(m-1)]$ are angles and r is the radius of a m-sphere. That is

$$r\ Exponential(\lambda)$$

$$\phi[j]\ Uniform(0, pi), j = 1...m - 2$$

$$\phi[m-1]\ Uniform(0, 2pi)$$

J_m is the Jacobian of this transformation

Value

a `cgeneric` object, see `INLAtools::cgeneric()` for details.

References

Lewandowski, Daniel, Dorota Kurowicka, and Harry Joe. 2009. "Generating Random Correlation Matrices Based on Vines and Extended Onion Method." Journal of Multivariate Analysis 100: 1989–2001.

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

Description

This creates an `cgeneric` (see `INLAtools::cgeneric()`) containing the necessary data to implement the penalized complexity prior for a correlation matrix considering a three as proposed in Sterrantino et. al. 2025 doi:10.1007/s1026002500788y.

Usage

```
cgeneric_treepcor(
  model,
  lambda,
  sigma.prior.reference,
  sigma.prior.probability,
  ...
)
```

Arguments

model	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.
...	additional arguments passed on to INLAtools::cgenericBuilder().

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

cgeneric/cgeneric object.

See Also

treepcor() and INLAtools::cgeneric()

---

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

Description

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

Usage

```
cgeneric_Wishart(n, dof, R, debug = FALSE, useINLAprecomp = TRUE, shlib = 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 'shlib' is provided.
shlib	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 cgeneric, INLAtools::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

fillLprec *Function to fill-in a Cholesky matrix*

Description

Function to fill-in a Cholesky matrix

Usage

```
fillLprec(L, lfi)
```

Arguments

L	matrix as the lower triangle containing the Cholesky decomposition of a initial precision matrix whose non-zeros are only at the position where the lower triangle side of the precision matrix is also non-zero
lfi	integer vector used as indicator of the position in the lower matrix where are the fill-in elements. Must be col then row ordered.

Value

lower triangular matrix with the filled-in elements thus Q0 can be computed.

graphpcor-class *graphpcor: correlation from nodes and edges*

Description

A graphpcor is a graph where a node represents a variable and an edge represent a conditional distribution. The correlation built from a graphpcor consider the parameters for the Cholesky of a precision matrix, whose non-zero pattern is given from the graph.

Usage

```
graphpcor(...)

## 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(x)

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

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

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

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

Arguments

...	a list of arguments
x	graphpcor
object	graphpcor object
which	not used
y	not used
model	graphpcor model object

Value

a graphpcor object

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
- vcov(graphpcor): The vcov method for a graphpcor

Functions

- `graphpcor()`: The `graphpcor` generic method for graphpcor
- `graphpcor(formula)`: Each term to represent a node, and each ~ to represent an edge.
- `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'
- `cgeneric(graphpcor)`: The `cgeneric` method for `graphpcor` uses `cgeneric_graphpcor()`
- `cgeneric(matrix)`: The `cgeneric` method for `matrix` uses `cgeneric_graphpcor()`

Examples

```
library(graphpcor)

par(mfrow = c(2, 3), mar = c(0,0,0,0))
plot(graphpcor(x~y+v, z~y+v))
plot(graphpcor(x~y,x~v,z~y,z~v))
plot(graphpcor(x~y, v~x, y~z, z~v))
plot(graphpcor(y~x, v~x, z~y, z~v))
plot(graphpcor(y~x+z, v~z+x))
plot(graphpcor(y~x+z, v~x, z~v))

## the graph in Example 2.6 of the GMRF book
g <- graphpcor(x ~ y + v, z ~ y + v)

class(g)
g

par(mfrow=c(1,1))
plot(g)

summary(g) ## the graph: nodes and edges (nodes ordered as given)

ne <- dim(g)
ne

## sometimes we need it
G <- Laplacian(g)
G

## alternatively
all.equal(G,
          Laplacian(graphpcor(x~y, v~x, y~z, z~v)))
all.equal(G,
          Laplacian(graphpcor(x~y, x~v, z~y, v~z)))
```

```
plot(graphpcor(G)) ## from a matrix

## base model (theta for lower triangle Cholesky)
theta0l <- rep(-0.5, ne[2])

## vcov() method for graphpcor computes the correlation
##  if only theta for lower of L is provided
C0 <- vcov(g, theta = theta0l)
C0

## the precision for a correlation matrix
Q0 <- prec(g, theta = theta0l)
Q0

all.equal(C0, as.matrix(solve(Q0)))

## the Hessian matrix around a base model
I0 <- basepcor(theta0l, p = ne[1], itheta = g)
I0

## a base model can also be a matrix
## however it shall give a precision with
## same sparse pattern as the graph
all.equal(I0, basepcor(C0, p = ne[1], itheta = g))

## the 'iid' case would be
vcov(g, theta = rep(0, ne[2]))
vcov(g, theta = rep(0, sum(ne)))

## marginal variance specified throught standard errors
sigmas <- c(0.3, 0.7, 1.2, 0.5)
## the covariance
vcov(g, theta = c(log(sigmas), rep(0, ne[2]))) ## IID
vcov(g, theta = c(log(sigmas), theta0l))

vcov(g, theta = rep(-3, ne[2])) ## no edge 2~3 but high correlation!!!
```

`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

---

KLD10 *Compute the KLD between two multivariate Gaussian distributions, assuming equal mean vector*

Description

Compute the KLD between two multivariate Gaussian distributions, assuming equal mean vector

Usage

```
KLD10(C1, C0, L1, L0)
```

Arguments

C1	is a correlation matrix.
C0	is a correlation matrix of the base model.
L1	is the Cholesky of C1.
L0	is the Cholesky of C0.

Details

By assuming equal mean vector we have

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

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(x)

## Default S3 method:
Laplacian(x)

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

Arguments

x	object defining a graph

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

Lprec0 *Compute the (lower triangle) Cholesky of the initial precision* `Q0`.

Description

Compute the (lower triangle) Cholesky of the initial precision `Q0`.

Usage

```
Lprec0(theta, p, itheta, d0)
```

Arguments

theta	numeric parameter vector.
p	integer (needed if base is vector): the dimention.
itheta	integer vector or 'graphpcor' to specify the (vectorized) position where 'theta' is placed in the initial (before the fill-in) Cholesky (lower triangle) factor. If missing, default, assumes the dense case as itheta = which(lower.tri(...)), giving length(theta)=p(p-1)/2.
d0	numeric vector to specify the diagonal of the Cholesky factor for the initial precision matrix Q0. Default, if not provided, is d0 = p:1.

Details

The (lower triangle) Cholesky factor of the initial precision for a correlation matrix contains the parameters in the non-zero elements of the lower triangle side of the precision matrix. The filled-in elements are computed from them using fillLprec().

Value

lower triangular matrix

---

param-utils *Internal functions to map between Euclidean and spherical coordinates*

Description

Internal functions to map between Euclidean and spherical coordinates

Usage

```
rphi2x(rphi)

x2rphi(x)

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

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

Arguments

rphi	numeric vector where the first element is the radius and the remaining are the angles
x	parameters in the Euclidean 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

Details

For details, please see the wikipedia entry on 'N-sphere' at N-sphere

Functions

- rphi2x(): Map between spherical to Euclidean coordinates
- x2rphi(): Transform from Euclidean coordinates to spherical
- rtheta(): Drawn samples from the PC-prior for correlation
- dtheta(): PC-prior density for the correlation matrix

---

treepcor-class *treepcor: correlation from tree*

Description

A tree with two kind of nodes, parents and children. The parents are nodes with children. The children are nodes with no children. This is used to model correlation matrices, where parents represent latent variables, and children represent the variables of interest.

Usage

```
treepcor(...)

## 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)

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

Arguments

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

Details

In the formula, the left side are parent variables, and the right side include all the children and parents that are also children. 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 parent variables are identified as `p1`, ..., `pm`, where the `m` is the number of parent variables. The main parent (first) should be identified as `p1`. Except `p1` all the other parent variables have an ancestor, which is a parent variable.

Value

a `treepcor` object

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

- `treepcor()`: A tree from a formula for each parent.
- `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`.
- `cgeneric(treepcor)`: The `cgeneric` method for `treepcor`, uses `cgeneric_treepcor()`

Examples

```
## for details see
## https://link.springer.com/article/10.1007/s10260-025-00788-y

library(graphpcor)

if(FALSE) {

### examples of what is not allowed
    treepcor(p1 ~ p2)
    treepcor(p1 ~ c2)

    treepcor(
        p1 ~ c1 + c2,
        p2 ~ c3)

    treepcor(
        p1 ~ c1 + c2,
        p2 ~ p1 + c2 + c3)

    treepcor(
        p1 ~ c1 + c2,
        p2 ~ p3 + c2 + c3)

    treepcor(
        p1 ~ p2 + c1 + c2,
        p2 ~ c2 + c3)

}

### allowed cases

## 3 children and 1 parent
g1 <- treepcor(p1 ~ c1 + c2 - c3)

g1
```

```
dim(g1)

summary(g1)

plot(g1)

prec(g1)

(q1 <- prec(g1, theta = c(0)))

v1 <- chol2inv(chol(q1))

v1

cov2cor(v1)

vcov(g1)
vcov(g1, theta = 0)
vcov(g1, theta = -1)
vcov(g1, theta = 1)

cov2cor(vcov(g1))
cov2cor(vcov(g1, theta = -1))
cov2cor(vcov(g1, theta = 1))

## 4 children and 2 parent
g2 <- treepcor(
    p1 ~ p2 + c1 + c2,
    p2 ~ c3 - c4)
g2
dim(g2)
summary(g2)

plot(g2)

prec(g2)
prec(g2, theta = c(0, 0))
prec(g2, theta = c(-1, 1))

cov2cor(solve(prec(g2, theta = c(0,0))))
vcov(g2, theta = c(0,0))
vcov(g2, theta = c(log(4:1), 0,0))

vcov(g2)

g2

## 4 children and 2 parent (notice the signs)
g3 <- treepcor(
    p1 ~ -p2 + c1 + c2,
    p2 ~ -c3 + c4)
g3
```

```
dim(g3)
summary(g3)

summary(g2)
summary(g3)

par(mfrow = c(1, 2), mar = c(0,0,0,0))
plot(g2)
plot(g3)

prec(g2)
prec(g3)

prec(g2, theta = c(0, 0))
prec(g3, theta = c(0, 0))

vcov(g2, theta = c(0, 0))
vcov(g3, theta = c(0, 0))

g3


drop1(g2)
drop1(g3)

prec(g3)
prec(drop1(g3))

n3 <- dim(g3)[1]
all.equal(
    solve(prec(g2, theta = c(0, 0)))[1:n3, 1:n3],
    solve(prec(g3, theta = c(0, 0)))[1:n3, 1:n3]
)

vcov(g2, theta = c(0,0))
vcov(g3, theta = c(0,0))
```

Index