

# Package ‘graphpcor’

May 7, 2026

**Type** Package

**Title** Models for Correlation Matrices Based on Graphs

**Version** 0.1.25

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

**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](https://doi.org/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

**NeedsCompilation** yes

**Depends** R (>= 4.3), Matrix, INLAtools (> 0.1.3), numDeriv

**Imports** methods, stats, utils, igraph

**Suggests** knitr, INLA (>= 24.02.09)

**BuildVignettes** true

**VignetteBuilder** knitr

**Config/roxygen2/version** 8.0.0

**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** 2026-05-07 13:40:07 UTC

**Contents**

basecor . . . . .	2
basecor-utils . . . . .	5
basepcor . . . . .	6
basepcor-utils . . . . .	9
cgeneric_graphpcor . . . . .	10
cgeneric_LKJ . . . . .	11
cgeneric_pc_correl . . . . .	12
cgeneric_treepcor . . . . .	14
cgeneric_Wishart . . . . .	15
dLKJ . . . . .	16
edges . . . . .	16
internal-utils . . . . .	17
Laplacian . . . . .	18
Laplacian.graphpcor . . . . .	19
numeric-utils . . . . .	21
param-utils . . . . .	22
stan_add . . . . .	23
treepcor . . . . .	25
<b>Index</b>	<b>29</b>

---

 basecor

*A base Cholesky model of a correlation matrix*


---

**Description**

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

**Usage**

```

basecor(base, p, parametrization = "cpc", iparams, iLtheta)

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

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

## 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(iLtheta)' 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".
iparams	integer ordered vector with length equal the number of parameters used to specify common parameter values. If missing, assumed to be 1:length(theta). Example: By setting iparams = c(1,1,2,3) the first and second parameters are considered to be the same.
iLtheta	integer vector to specify the (vectorized) position where 'theta' will be placed in the (lower triangle) Cholesky factorization of the correlation matrix. Default (missing, or if NULL) and assumes iLtheta = 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  $k = 1, \dots, m$ , and the two  $p \times p$  matrices

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

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 matrix 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  $k = 1, \dots, m$ , 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,  $\mathbf{I}\theta$ , 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

## Methods (by class)

- `basecor(numeric)`: Build a basecor from the parameter vector.
- `basecor(matrix)`: Build a basecor from a correlation matrix.

## Methods (by generic)

- `print(basecor)`: Print method for 'basecor'

## Functions

- `basecor()`: Build a basecor 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>

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>

**Examples**

```

## A correlation matrix
c0 <- matrix(c( 1.0,  0.8, -0.5,
               0.8,  1.0, -0.4,
               -0.5, -0.4,  1.0), 3)

## build the 'basecor'
b3 <- basecor(c0) ## base as matrix
b3

## elements
b3$base
b3$theta

## from 'theta'
th3 <- b3$theta
bth3 <- basecor(th3, p = 3) ## base as vector
bth3

## numerically the same
all.equal(c0, bth3$base, tol = 1e-4)

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

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

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

## Hessian around the base
hessian(b3)
hessian(b3, ifixed = 3)
hessian(b2)

```

---

basecor-utils

*Functions used by/to basecor*


---

**Description**

Functions used by/to basecor

Map a correlation matrix to theta.

**Usage**

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

```
corr2theta(corr)
```

**Arguments**

<code>theta</code>	numeric parameter vector.
<code>p</code>	integer with the dimension, the number of rows/columns of the correlation matrix.
<code>iLtheta</code>	integer vector to specify the (vectorized) position where 'theta' will be placed in the (lower triangle) Cholesky factorization of the correlation matrix. Default (missing, or if NULL) and assumes <code>iLtheta = which(lower.tri(...))</code> .
<code>parametrization</code>	character to specify the parametrization used. The available ones are "cpc" (or "CPC") or "sap" (or "SAP"). See Details. The default is "cpc".
<code>corr</code>	matrix as a correlation matrix

**Value**

matrix with lower triangle as the Cholesky factor of a correlation matrix, and attributes: `parametrization` ("cpc" or "sap"), `"theta"` the parameter vector under the parametrization, `"iLtheta"` the index in the lower triangle matrix for theta, `"logDeterminant"` the log determinant for the correlation matrix.

**Functions**

- `cholcor()`: Build the (lower) Cholesky for a correlation matrix
- `corr2theta()`: This function takes a correlation matrix and return the parameter vector that generates it with the canonical partial correlation parametrization.

---

 basepcor

---

*A base precision's Cholesky model for a correlation matrix*


---

**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 `iLtheta` for theta in the (lower) Cholesky, and the Hessian around it `I0`, see details.

**Usage**

```
basepcor(base, p, iLtheta, d0, iparams)

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

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

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

**Arguments**

base	a correlation matrix, or numeric vector as the parameter(s) to define correlation matrix.
p	integer (needed if base is vector): the dimension.
iltheta	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 <code>iltheta = which(lower.tri(...))</code> , giving <code>length(theta)=p(p-1)/2</code> .
d0	numeric vector to specify the diagonal of the Cholesky factor for the initial precision matrix $Q_0$ . Default, if not provided, is <code>d0 = p:1</code> .
iparams	integer ordered vector with length equal the number of parameters used to specify common parameter values. Default is <code>1:m</code> , <code>m=length(theta)</code> . Example: By setting <code>iparams = c(1, 1, 2, 3)</code> , <code>m=3</code> , the first and second parameters are considered to be the same. NOTE: <code>c(1, 2, 1)</code> is allowed, but <code>c(2, 1, 2)</code> is not.
x	a basepcor object.
...	further arguments passed on.

**Details**

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

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

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,  $\mathbf{I}\theta$ , 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

**Methods (by class)**

- `basepcor(numeric)`: Build a basepcor from the parameter vector.
- `basepcor(matrix)`: Build a basepcor from a correlation matrix.

**Methods (by generic)**

- `print(basepcor)`: Print method for 'basepcor'

## Functions

- `basepcor()`: Build a `basepcor` object.

## Examples

```
## A correlation matrix
c0 <- matrix(c( 1.0,  0.8, -0.5,
               0.8,  1.0, -0.4,
               -0.5, -0.4,  1.0), 3)

## p = 3, m = 3
b1 <- basepcor(c0)
b1

## p = 3, m = 2
b2 <- basepcor(c0, iLtheta = c(2,3))
b2

all.equal(b1$base, b2$base)

## Hessian
hessian(b2)
hessian(b1, ifixed = 3)
hessian(b1)

## p = 4, m = 4
th4 <- c(0.5,-1,0.5,-0.3)
ith4 <- c(2,3,8,12)
b44 <- basepcor(th4, p = 4, iLtheta = ith4)
b44

Sparse(round(solve(b44$base), 4), zeros.rm = TRUE)

## p = 4, m = 3 (with some common theta)
th3 <- c(0.5, -1, -0.3)
ip3 <- c(1, 2, 1, 3) ## 1st == 3rd
b43 <- basepcor(th3, p = 4, iLtheta = ith4, iparams = ip3)

all.equal(b44$base, b43$base) ## TRUE

## parameter dimension is now reduced
hessian(b44)
hessian(b43)

## If a subset of the parameters are known (fixed), then the
## Hessian is only computed with respect to the unknown ones
hessian(basepcor(th4, p=4, iLtheta = ith4), ifixed = 2:3)
hessian(basepcor(th4, p=4, iLtheta = ith4), ifixed = 1)
hessian(basepcor(th4, p=4, iLtheta = ith4), ifixed = 3)

hessian(basepcor(th3, p=4, iLtheta = ith4,
                 iparams = ip3), ifixed = 3)
```

```

hessian(basepcor(th3, p=4, iLtheta = ith4,
                 iparams = ip3), ifixed = 2:3)
hessian(basepcor(th3, p=4, iLtheta = ith4,
                 iparams = ip3), ifixed = 1:2)

## check
hessian(basepcor(th3, p=4, iLtheta = ith4,
                 iparams = ip3), ifixed = NULL)

```

---

basepcor-utils                      *Functions used by/to basepcor*

---

## Description

Functions used by/to basepcor

## Usage

```

fillLprec(L, lfi)

Lprec0(theta, p, iLtheta, d0)

```

## 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.
theta	numeric, the parameter vector.
p	integer (needed if base is vector): the dimension.
iLtheta	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 <code>iLtheta = which(lower.tri(...))</code> , giving <code>length(theta)=p(p-1)/2</code> .
d0	numeric vector to specify the diagonal of the Cholesky factor for the initial precision matrix $Q_0$ . Default, if not provided, is <code>d0 = p:1</code> .

## 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 with the filled-in elements thus  $Q\theta$  can be computed.

lower triangular matrix

**Functions**

- `fillLprec()`: Function to fill-in a Cholesky matrix
- `Lprec $\theta$ ()`: Compute the (lower triangle) Cholesky of the initial precision  $Q\theta$ .

---

`cgeneric_graphpcor`      *Build an cgeneric for a (graph based) correlation matrix PC-prior.*

---

**Description**

From either a `graphpcor` (see `graphpcor()`) 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,
  iparams,
  cfixed,
  d $\theta$ ,
  ...
)
```

**Arguments**

<code>model</code>	a <code>graphpcor</code> (see <code>graphpcor()</code> ) or a square matrix (to be used as a graph) to define the precision structure of the model.
<code>lambda</code>	the parameter for the exponential prior on the radius of the sphere, see details in the PC-multivariate vignette.
<code>base</code>	numeric vector, correlation matrix or <code>basepcor</code> object. See <code>basepcor()</code> for details. If the output of a <code>basepcor()</code> is provided, <code>iLtheta</code> will be considered from it, and <code>iparams</code> for the correlation parameters as well.
<code>sigma.prior.reference</code>	numeric vector to set the reference for each standard deviation parameter for its PC-prior. If missing, the model will be assumed as for a correlation. Note: <code>iparams</code> will be applied here as <code>sigma.prior.reference[iparams[1:n]]</code> .

<code>sigma.prior.probability</code>	numeric vector with to set the probability statement of the PC prior for each marginal variance parameters. The probability statement is $P(\text{sigma} < \text{sigma.prior.reference}) = p$ . If missing, all the marginal variances are considered as known. If a vector is given and a probability is NA, 0 or 1, the corresponding <code>sigma.prior.reference</code> will be used as fixed. Note: <code>iparams</code> will be applied here as <code>sigma.prior.probability[iparams[1:n]]</code> .
<code>iparams</code>	integer vector with length equal $n+m$ , where $m$ is the number of correlation parameters, to identify (possible) common parameters in the model. Default is $1:(n+m)$ . Note: <code>c(1, 2, 1)</code> is allowed, but <code>c(2, 1, 2)</code> is not.
<code>cfixed</code>	integer vector to specify which correlation parameters are treated as known and fixed. By default all correlation parameters are treated as unknown. Example: if <code>cfixed = c(1, 3)</code> , the first and third correlation parameters will be treated as fixed and the Hessian will be computed for the second correlation parameter. Please see the examples in <a href="#">basepcor()</a> . Note: consider <code>iparams[n+1:m]-iparams[n]</code> .
<code>d0</code>	numeric vector to specify the diagonal of the Cholesky factor for the initial precision matrix $Q_0$ , passed on to <a href="#">basepcor()</a> . Default is <code>d0 = n:1</code> .
<code>...</code>	additional arguments passed on to <a href="#">INLAtools::cgeneric()</a> , such as <code>debug</code> , <code>shlib</code> and <code>useINLApcomp</code> .

### Details

The parametrization is set as in [basepcor\(\)](#) and the base is used to define an informative prior, as derived in the `pcmultivariate` vignette.

### Value

`cgeneric` object.

### See Also

[graphpcor\(\)](#) and [basepcor\(\)](#)

---

`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 $P(\text{sigma} < \text{sigma.prior.reference}) = p$ . 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 <code>INLAtools::cgeneric()</code> .

**Value**

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

**See Also**

It uses the Canonical Partial Correlation (CPC), parametrization, see `basecor()` for details.

---

cgeneric\_pc\_correl      *Build an cgeneric for a correlation matrix PC-prior.*

---

**Description**

Build an cgeneric for a correlation matrix PC-prior.

**Usage**

```
cgeneric_pc_correl(n, base, iltheta, iparams, ...)

## S3 method for class 'basecor'
cgeneric(
  model,
  lambda,
  sigma.prior.reference,
  sigma.prior.probability,
  iparams,
  cfixed,
  ...
)
```

**Arguments**

n	integer to define the size of the matrix, same as p in <code>basecor()</code> .
base	numeric vector, matrix or <code>basecor</code> to define the base correlation model. See <code>basecor()</code> for details. If the output of a <code>basecor()</code> is provided, <code>iltheta</code> and <code>iparams</code> (for the correlation parameters) will be considered from this.
iltheta	integer vector to specify the (vectorized) position where 'theta' will be placed in the (lower triangle) Cholesky factorization of the correlation matrix.
iparams	integer vector with length equal $n+m$ , where $m$ is the number of correlation parameters, to identify (possible) common parameters in the model. Default is $1:(n+m)$ . Note: <code>c(1, 2, 1)</code> is allowed, but <code>c(2, 1, 2)</code> is not.
...	additional arguments passed on to <code>INLAtools::cgeneric()</code> , such as <code>debug</code> , <code>shlib</code> and <code>useINLAPrecomp</code> .
model	a <code>basecor</code> object.
lambda	the parameter for the exponential prior on the radius of the sphere, see details in the PC-multivariate vignette.
sigma.prior.reference	numeric vector to set the reference for each standard deviation parameter for its PC-prior. If missing, the model will be assumed as for a correlation. Note: <code>iparams</code> will be applied here as <code>sigma.prior.reference[iparams[1:n]]</code> .
sigma.prior.probability	numeric vector with to set the probability statement of the PC prior for each marginal variance parameters. The probability statement is $P(\text{sigma} < \text{sigma.prior.reference}) = p$ . If missing, all the marginal variances are considered as known. If a vector is given and a probability is NA, 0 or 1, the corresponding <code>sigma.prior.reference</code> will be used as fixed. Note: <code>iparams</code> will be applied here as <code>sigma.prior.probability[iparams[1:n]]</code> .
cfixed	integer vector to specify which correlation parameters are treated as known and fixed. By default all correlation parameters are treated as unknown. Example: if <code>cfixed = c(1, 3)</code> , the first and third correlation parameters will be treated as fixed and the Hessian will be computed for the second correlation parameter. Please see the examples in <code>basepcor()</code> . Note: consider <code>iparams[n+1:m]-iparams[n]</code> .

**Details**

The parametrization is set as in `basecor()` and the base is used to define an informative prior, as derived in the `pcmultivariate` vignette.

**Value**

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

**Functions**

- `cgeneric(basecor)`: Build a `cgeneric` for a `basecor`.

## References

Daniel Simpson, Håvard Rue, Andrea Riebler, Thiago G. Martins and Sigrunn H. Sørbye (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\_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](https://doi.org/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 <a href="#">INLAtools::cgenericBuilder()</a> .

## 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**

[treecor\(\)](#) and [INLAtools::cgeneric\(\)](#)

---

cgeneric_Wishart	<i>Build an cgeneric to implement the Wishart prior for a precision matrix.</i>
------------------	---

---

**Description**

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

**Usage**

```
cgeneric_Wishart(n, dof, R, ...)
```

**Arguments**

n	integer to define the size of the precision matrix
dof	degrees of freedom model parameter
R	lower triangle of the scale matrix parameter
...	additional arguments passed on to <a href="#">INLAtools::cgeneric()</a> , such as debug, shlib and useINLAPrecomp.

**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

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

**Value**

a cgeneric, [cgeneric\(\)](#) object.

---

dLKJ	<i>The LKJ density for a correlation matrix</i>
------	---

---

**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

---

edges	<i>Retrieve edges of an object</i>
-------	------------------------------------

---

**Description**

Retrieve edges of an object

**Usage**

```
edges(object)

## Default S3 method:
edges(object)

## S3 method for class 'graphpcor'
edges(object)

## S3 method for class 'treepcor'
edges(object)
```

**Arguments**

object	graphpcor object
--------	------------------

**Methods (by class)**

- `edges(default)`: Default method for edges
- `edges(graphpcor)`: Extract the edges of a graphpcor.
- `edges(treepcor)`: Extract the edges of a treepcor

---

internal-utils      *Internal functions.*

---

**Description**

Internal functions.

**Usage**

```
p_iLtheta_fncheck(p, iLtheta)
```

```
m_iparams_fncheck(m, iparams)
```

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

```
pcSigmasCheck(nsigmas, sigma.prior.reference, sigma.prior.probability)
```

**Arguments**

<code>p</code>	integer (needed if base is vector): the dimension.
<code>iLtheta</code>	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 <code>iLtheta = which(lower.tri(...))</code> , giving <code>length(theta)=p(p-1)/2</code> .
<code>m</code>	integer to specify the number of parameters
<code>iparams</code>	integer ordered vector with length equal the number of parameters used to specify common parameter values. Default is <code>1:m</code> , <code>m=length(theta)</code> . Example: By setting <code>iparams = c(1, 1, 2, 3)</code> , <code>m=3</code> , the first and second parameters are considered to be the same. NOTE: <code>c(1, 2, 1)</code> is allowed, but <code>c(2, 1, 2)</code> is not.
<code>C1</code>	is a correlation matrix.
<code>C0</code>	is a correlation matrix of the base model.
<code>L1</code>	is the Cholesky of <code>C1</code> .
<code>L0</code>	is the Cholesky of <code>C0</code> .
<code>nsigmas</code>	number of parameters.
<code>sigma.prior.reference</code>	numeric vector to set the reference for each standard deviation parameter for its PC-prior.

`sigma.prior.probability`

numeric vector with to set the probability statement of the PC prior for each marginal variance parameters. The probability statement is  $P(\text{sigma} < \text{sigma.prior.reference}) = p$ . If missing, all the marginal variances are considered as known. If a vector is given and a probability is NA, 0 or 1, the corresponding `sigma.prior.reference` will be used as fixed.

## Details

By assuming equal mean vector we have

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

## Functions

- `p_iLtheta_fncheck()`: Function to deal with `p` and `iLtheta`
- `m_iparams_fncheck()`: Function to deal with `m` and `iparams`
- `KLD10()`: Compute the KLD between two multivariate Gaussian distributions, assuming equal mean vector
- `pcSigmasCheck()`: Check the PC-prior arguments for `sigma`.

---

Laplacian

*The Laplacian of a graph*

---

## Description

The (symmetric) Laplacian of a graph is a square matrix with dimension 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  $i \sim j$  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)`

## 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
- Laplacian(Matrix): The Laplacian of a Matrix

---

Laplacian.graphpcor    *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**

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

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

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

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

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

```
## S3 method for class 'matrix'
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, ...)

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

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

gphcQ(model, ...)

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

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

### Arguments

x	graphpcor
...	matrix or Matrix (treated as binary) or a vector or list of formula (or character interpreted as formula)
object	graphpcor
y	not used
model	graphpcor model object

### Methods (by class)

- `graphpcor(list)`: Each element may be a character or formula.
- `graphpcor(formula)`: Each term represents a node, and each ~ an edge.
- `graphpcor(character)`: Each term represents a node, and each ~ an edge.
- `graphpcor(matrix)`: Build a graphpcor from a matrix object
- `graphpcor(Matrix)`: Build a graphpcor for a Matrix object

### Methods (by generic)

- `Laplacian(graphpcor)`: The Laplacian method for a graphpcor
- `print(graphpcor)`: The print method for graphpcor
- `summary(graphpcor)`: The summary method for graphpcor
- `dim(graphpcor)`: The dim method for graphpcor
- `plot(graphpcor)`: The plot method for a graphpcor
- `vcov(graphpcor)`: The vcov method for a graphpcor
- `cgeneric(graphpcor)`: The cgeneric method for graphpcor uses [cgeneric\\_graphpcor\(\)](#)

**Functions**

- `gphcQ()`: The precision method for 'graphpcor'
- `cgeneric(Matrix)`: The `cgeneric` method for `Matrix` uses `cgeneric_graphpcor()`

---

 numeric-utils

*Functions for numerical algorithms*


---

**Description**

Functions for numerical algorithms

**Usage**

```
dspd(M, decomposition = "svd")

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

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

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

**Arguments**

<code>M</code>	square matrix.
<code>decomposition</code>	character to inform which decomposition is to be applied to the hessian. The options are "eigen", "svd" and "chol". Default is "svd".
<code>func</code>	model object definition for a correlation matrix.
<code>x</code>	for a <code>graphpcor</code> it is the parameter vector, otherwise not used.
<code>method</code>	see <code>numDeriv::hessian()</code>
<code>method.args</code>	see <code>numDeriv::hessian()</code>
<code>...</code>	used to pass <code>ifixed</code> , an integer vector to indicate model parameters as fixed. If not used, all parameters are treated unknown.

**Value**

`dspd` returns a list with the decomposition elements, "logDeterminant" (of the original matrix), "sqrt" (its 'square root') and "sqrtInv" (its inverse 'square root'). `hessian` returns the Hessian matrix with the Hessian

**Functions**

- `dspd()`: `hessian`: Evaluate the Hessian of the KLD for a `basecor`. `spd`: decompose a positive definite matrix, and compute useful elements out of that.
- `hessian(basecor)`: Evaluate the Hessian for a `basecor`.
- `hessian(basepcor)`: Evaluate the hessian of the KLD for a `basepcor`.
- `hessian(graphpcor)`: Evaluate the hessian of the KLD for a `graphpcor` correlation model around a base model.

---

 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**

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

**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

---

 stan\_add

*Build/add STAN code/data for the correlation's PC-prior.*


---

**Description**

Build/add STAN code/data for the correlation's PC-prior.

**Usage**

```
stan_add(x, model, lambda, name)
```

```
stan_add_pc_correl(x, model, lambda, name)
```

```
stan_add_graphpcor(x, model, lambda, name)
```

```
stan_add_code(x, to_add)
```

**Arguments**

<code>x</code>	either a STAN code or list with the data used to fit a STAN model.
<code>model</code>	either a character ("pc_correl" or "graphcor") or a <code>basecor</code> or a <code>basepcor</code> object to define the base correlation model. See <a href="#">basecor()</a> or <a href="#">basepcor()</a> .
<code>lambda</code>	the parameter for the exponential prior on the radius of the sphere, see details in the PC-multivariate vignette.
<code>name</code>	character to provide the name for the Cholesky of a correlation matrix or the correlation matrix. See details.
<code>to_add</code>	named list with the code to be added

**Details**

The parametrization is set as in [basecor\(\)](#) or [basepcor\(\)](#). If a `basecor` is provided, the prior would be considered for the Cholesky factor of a correlation matrix. If a `basepcor` is provided, the prior would be considered for a correlation matrix (parametrized from its inverse). The base is used to define an informative prior, as derived in the `pcmultivariate` vignette.

**Value**

a list of two elements, one as a list of three additional code to be added into a STAN code and the other with the required additional data.

## Functions

- `stan_add_pc_correl()`: method for basecor
- `stan_add_graphpcor()`: method for basepcor
- `stan_add_code()`: add code at the end of each section

## References

Daniel Simpson, Håvard Rue, Andrea Riebler, Thiago G. Martins and Sigrunn H. Sørbye (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>

## Examples

```
## STAN model code without the prior for L, as LCorr
Scode0 <- "
data {
  int<lower=1> p;
  int<lower=1> n;
  vector[p] y[n];
  vector[p] mu;
}
model {
  y ~ multi_normal_cholesky(mu, LCorr);
}
generated quantities {
  corr_matrix[p] Corr = tcrossprod(LCorr);
}
"

## add the pc_correl code
Scode <- stan_add(Scode0, 'pc_correl', lambda = 1, name = "LCorr")
Scode

Sdata0 <- list(
  p = as.integer(3),
  n = as.integer(1),
  y = matrix(0,1,3),
  mu = rep(0.0, 3)
)

baseC <- basecor(rep(0,3), 3)

Sdata <- stan_add(Sdata0, baseC, lambda = 1, name = "LCorr")
str(Sdata)
```

---

treepcor	<i>treepcor: correlation from tree</i>
----------	--

---

## 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. A tree is defined from a formula for each parent.

## Usage

```
treepcor(...)  
  
## S3 method for class 'treepcor'  
print(x, ...)  
  
## S3 method for class 'treepcor'  
summary(object, ...)  
  
## S3 method for class 'treepcor'  
dim(x, ...)  
  
## S3 method for class 'treepcor'  
drop1(object, scope, ...)  
  
## S3 method for class 'treepcor'  
plot(x, y, ...)  
  
tpcQ(model, ...)  
  
etreepcor2precision(d.el)  
  
## S3 method for class '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

scope	not used
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  $c_1, \dots, c_n$ , where  $n$  is the total number of children variables. The parent variables are identified as  $p_1, \dots, p_m$ , where the  $m$  is the number of parent variables. The main parent (first) should be identified as  $p_1$ . Except  $p_1$  all the other parent variables have an ancestor, which is a parent variable.

### Value

a treepcor object

### Methods (by generic)

- `print(treepcor)`: The print method for a treepcor
- `summary(treepcor)`: The summary method for a treepcor
- `dim(treepcor)`: The dim for a treepcor
- `drop1(treepcor)`: The drop1 method for a treepcor
- `plot(treepcor)`: The plot method for a treepcor
- `vcov(treepcor)`: The vcov method for a treepcor
- `cgeneric(treepcor)`: The cgeneric method for treepcor, uses `cgeneric_treepcor()`

### Functions

- `tpcQ()`: The precision 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.

### Examples

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

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  
tree1 <- treepcor(p1 ~ c1 + c2 - c3)  
  
tree1  
  
dim(tree1)  
  
summary(tree1)  
  
plot(tree1)  
  
tpcQ(tree1)  
  
(q1 <- tpcQ(tree1, theta = c(0)))  
  
v1 <- chol2inv(chol(q1))  
  
v1  
  
cov2cor(v1)  
  
vcov(tree1, raw = TRUE)  
cov2cor(vcov(tree1, raw = TRUE))  
vcov(tree1)  
  
vcov(tree1, theta = 0)  
vcov(tree1, theta = -1)  
vcov(tree1, theta = 1)  
  
cov2cor(vcov(tree1))  
cov2cor(vcov(tree1, theta = -1))  
cov2cor(vcov(tree1, theta = 1))
```

```

## 4 children and 2 parent
tree2 <- treepcor(
  p1 ~ p2 + c1 + c2,
  p2 ~ -c3 + c4)
tree2
dim(tree2)
summary(tree2)

tpcQ(tree2)
tpcQ(tree2, theta = c(0, 0))
tpcQ(tree2, theta = c(-1, 1))

vcov(tree2, raw = TRUE)
cov2cor(vcov(tree2, raw = TRUE))

cov2cor(solve(tpcQ(tree2, theta = c(0,0))))
vcov(tree2, theta = c(0,0))
vcov(tree2, theta = c(log(4:1), 0,0))

vcov(tree2)

tree2

## 4 children and 2 parent (notice the signs)
tree2b <- treepcor(
  p1 ~ -p2 + c1 + c2,
  p2 ~ -c3 + c4)
tree2b
dim(tree2b)
summary(tree2b)

summary(tree2)
summary(tree2b)

par(mfrow = c(1, 2), mar = c(0,0,0,0))
plot(tree2)
plot(tree2b)

tpcQ(tree2)
tpcQ(tree2b)

## prec is not equal
all.equal(tpcQ(tree2, theta = c(0, 0)),
          tpcQ(tree2b, theta = c(0, 0)))

## vcov is equal
all.equal(vcov(tree2, theta = c(0, 0)),
          vcov(tree2b, theta = c(0, 0)))

```

# Index

basecor, 2  
basecor(), 12, 13, 23  
basecor-utils, 5  
basepcor, 6  
basepcor(), 10, 11, 13, 23  
basepcor-utils, 9

cgeneric(), 13, 15  
cgeneric.basecor (cgeneric\_pc\_correl), 12  
cgeneric.graphpcor (Laplacian.graphpcor), 19  
cgeneric.Matrix (Laplacian.graphpcor), 19  
cgeneric.treepcor (treepcor), 25  
cgeneric\_graphpcor, 10  
cgeneric\_graphpcor(), 20, 21  
cgeneric\_LKJ, 11  
cgeneric\_pc\_correl, 12  
cgeneric\_treepcor, 14  
cgeneric\_treepcor(), 26  
cgeneric\_Wishart, 15  
cholcor (basecor-utils), 5  
corr2theta (basecor-utils), 5

dim.graphpcor (Laplacian.graphpcor), 19  
dim.treepcor (treepcor), 25  
dLKJ, 16  
drop1.treepcor (treepcor), 25  
dspd (numeric-utils), 21  
dtheta (param-utils), 22

edges, 16  
etreepcor2precision (treepcor), 25  
etreepcor2variance (treepcor), 25

fillLprec (basepcor-utils), 9  
fillLprec(), 9

gphcQ (Laplacian.graphpcor), 19  
graphpcor (Laplacian.graphpcor), 19  
graphpcor(), 10, 11

hessian.basecor (numeric-utils), 21  
hessian.basepcor (numeric-utils), 21  
hessian.graphpcor (numeric-utils), 21

INLAtools::cgeneric(), 10–15  
INLAtools::cgenericBuilder(), 14  
internal-utils, 17

KLD10 (internal-utils), 17

Laplacian, 18  
Laplacian.graphpcor, 19  
Lprec0 (basepcor-utils), 9

m\_iparams\_fncheck (internal-utils), 17

numDeriv::hessian(), 21  
numeric-utils, 21

p\_iltheta\_fncheck (internal-utils), 17  
param-utils, 22  
pcSigmasCheck (internal-utils), 17  
plot.graphpcor (Laplacian.graphpcor), 19  
plot.treepcor (treepcor), 25  
print.basecor (basecor), 2  
print.basepcor (basepcor), 6  
print.graphpcor (Laplacian.graphpcor), 19  
print.treepcor (treepcor), 25

rphi2x (param-utils), 22  
rtheta (param-utils), 22

stan\_add, 23  
stan\_add\_code (stan\_add), 23  
stan\_add\_graphpcor (stan\_add), 23  
stan\_add\_pc\_correl (stan\_add), 23  
summary.graphpcor (Laplacian.graphpcor), 19

`summary.treepcor (treepcor)`, [25](#)

`tpcQ (treepcor)`, [25](#)

`treepcor`, [25](#)

`treepcor()`, [14](#), [15](#)

`vcov.graphpcor (Laplacian.graphpcor)`, [19](#)

`vcov.treepcor (treepcor)`, [25](#)

`x2rphi (param-utils)`, [22](#)