Package 'gmat'

December 3, 2018

Type Package

Title Simulation of Graphically Constrained Matrices

Version 0.2.0

Date 2018-11-30

Encoding UTF-8

Description Simulation of positive definite matrices constrained by an undirected or acyclic directed graph structure.

License GPL (>= 2)

RoxygenNote 6.1.1

ByteCompile true

URL https://github.com/irenecrsn/gmat

BugReports https://github.com/irenecrsn/gmat/issues

Suggests testthat, covr

Imports igraph, stats

NeedsCompilation yes

Author Irene Córdoba [aut, cre] (<https://orcid.org/0000-0002-3252-4234>),
Gherardo Varando [aut] (<https://orcid.org/0000-0002-6708-1103>),
Concha Bielza [ths] (<https://orcid.org/0000-0001-7109-2668>),
Pedro Larrañaga [ths] (<https://orcid.org/0000-0003-0652-9872>)

Maintainer Irene Córdoba <irene.cordoba@upm.es>

Repository CRAN

Date/Publication 2018-12-03 13:02:39 UTC

R topics documented:

anti_t 2
dag-constrained correlation matrices 2
gmat 4
metropolis-hastings sampling 4

rgraph 6
set_cond_number 6
ug-constrained covariance matrices 7
vectorize 8

Index **9**

`anti_t` *Compute the anti transpose of a matrix (transpose with respect to the off-diagonal)*

Description

Compute the anti transpose of a matrix (transpose with respect to the off-diagonal)

Usage

```
anti_t(m)
```

Arguments

`m`	square matrix to compute the anti transpose

Value

The anti-transpose of m

`dag-constrained correlation matrices` *Simulation of correlation matrices*

Description

Sample correlation matrices, possibly with a zero pattern in its Cholesky decomposition constrained by an acyclic digraph.

Usage

```
chol_mh(N = 1, p = 3, d = 1, dag = NULL, ...)

chol_iid(N = 1, p = 3, d = 1, dag = NULL)

chol_polar(N = 1, p = 3, d = 1, dag = NULL, comp = "numeric")
```

Arguments

N	Number of samples.
p	Matrix dimension. Ignored if dag is provided.
d	Number in [0,1], the proportion of non-zero entries in the Cholesky factor of the sampled matrices. Ignored if dag is provided. Ignored by chol_polar()
dag	An igraph acyclic digraph specifying the zero pattern in the upper Cholesky factor of the sampled matrices. Nodes must be in ancestral order, with the first one having no parents.
...	Additional parameters for mh_u().
comp	String one of "numeric" or "recursive", indicating the computational method to use for sampling the angles for "unifconc" method

Details

Function chol_mh() uses the method described in Córdoba et al. (2018) and implemented in mh_u(), based on a Metropolis-Hastings algorithm over the upper Cholesky factorization.

The entries in the upper Cholesky factor are sampled i.i.d. by function chol_iid(), following Kalisch and Buhlmann (2007).

Function chol_polar() reparametrizes the Cholesky factor following the approach by Pourahmadi and Wang (2015), adapted to sample the upper Cholesky factor instead of the lower one.

Value

A three-dimensional array of length p x p x N

References

Córdoba I., Varando G., Bielza C., Larrañaga P. A fast Metropolis-Hastings method for generating random correlation matrices. *Lecture Notes in Computer Science* (IDEAL 2018), vol 11314, pp. 117-124, 2018.

Kalisch, M., Buhlmann, P. Estimating high-dimensional directed acyclic graphs with the PC-algorithm, *Journal of Machine Learning Research*, 8:613-636, 2007.

Pourahmadi, M., Wang, X. Distribution of random correlation matrices: Hyperspherical parameterization of the Cholesky factor, *Statistics & Probability Letters*, 106:5-12, 2015.

Examples

```
## Cholesky sampling via Metropolis-Hastings
# Generate a full matrix (default behaviour)
chol_mh()

# Generate a matrix with a percentage of zeros
chol_mh(d = 0.5)

# Generate a random acyclic digraph structure
dag <- rgraph(p = 3, d = 0.5, dag = TRUE)
igraph::print.igraph(dag)
```

```
# Generate a matrix complying with the predefined zero pattern
chol_mh(dag = dag)
## Cholesky sampling via i.i.d. Cholesky factor
# Generate a full matrix (default behaviour)
chol_iid()

# Generate a matrix with a percentage of zeros
chol_iid(d = 0.5)

# Generate a matrix complying with the predefined zero pattern
igraph::print.igraph(dag)
chol_iid(dag = dag)
## Cholesky sampling via polar parametrization of the upper Cholesky factor
# Generate a full matrix (default behaviour)
chol_polar()

# Generate a matrix with a percentage of zeros
chol_polar(d = 0.5)

# Generate a matrix complying with the predefined zero pattern
igraph::print.igraph(dag)
chol_polar(dag = dag)

# Performance comparison of numeric vs recursive integral (full matrix)
system.time(chol_polar(N = 10, p = 5))
system.time(chol_polar(N = 10, p = 5, comp = "recursive"))
```

`gmat` *gmat: Graphically constrained matrices.*

Description

gmat: Graphically constrained matrices.

`metropolis-hastings sampling`
Upper Cholesky factor sampling using Metropolis-Hastings

Description

Metropolis-Hasting algorithms to sample the upper Cholesky factor, using positive hemispheres of different dimensions. A zero pattern may be specified using an acyclic digraph.

Usage

```
mh_u(N = 1, p = 3, dag = NULL, h = 100, eps = 0.1)

mh_sphere(N = 1, k, i = 1, h = 100, eps = 0.01)
```

Arguments

N	Number of samples.
p	Dimension of the upper Cholesky factor.
dag	An igraph acyclic digraph specifying the zero pattern in the upper Cholesky factor of the sampled matrices. Nodes must be in ancestral order, with the first one having no parents.
h	Heating phase size for `mh_sphere()`.
eps	Perturbation variance for `mh_sphere()`.
k	Dimension of the hemisphere from which the sample is taken.
i	Integer, power of the first coordinate in the density.

Details

Function `mh_u()` returns a sample of `N` upper Cholesky factors whose rows have been generated using `mh_sphere()`. The dimensions of the hemispheres used to sample vary depending both on the row number of the Cholesky factor, and whether there is a zero pattern specified by `dag`.

The details of the algorithm implemented by `mh_sphere()` can be found in the paper Córdoba et al. (2018), including a discussion on theoretical convergence and numerical experiments for choosing its hyper parameters `h` and `eps`.

Author(s)

Gherardo Varando <gherardo.varando@math.ku.dk>

References

Córdoba I., Varando G., Bielza C., Larrañaga P. A fast Metropolis-Hastings method for generating random correlation matrices. *Lecture Notes in Computer Science* (IDEAL 2018), vol 11314, pp. 117-124, 2018.

Examples

```
## Upper Cholesky factor sampling
# Generate a random acyclic digraph
dag <- rgraph(p = 3, d = 0.5, dag = TRUE)
igraph::print.igraph(dag)

# Generate an upper Cholesky factor complying with such zero pattern
mh_u(dag = dag)
# We may also generate it with no zero pattern (full upper triangular)
mh_u()
## Hemisphere sampling
# 3D hemisphere from a density proportional to the square of the first coordinate
mh_sphere(N = 4, k = 3, i = 2)
```

rgraph *Random generation of acyclic digraphs and undirected graphs*

Description

Wrapper of functionality from package `igraph` for random generation of graphs.

Usage

```
rgraph(p, d, dag = FALSE)
```

Arguments

`p`	Number of vertices of the sampled graph
`d`	Proportion of edges in the generated graph
`dag`	Whether the generated graph should be acyclic directed

Details

When `dag = FALSE`, the graph is sampled from an Erdos-Renyi model. In the case where `dag = TRUE`, the upper triangle of the adjacency matrix of an Erdos-Renyi model is taken as the adjacency matrix for the acyclic digraph. This preserves the proportion of edges `d`.

Value

g The generated graph. If `dag = TRUE`, the nodes follow the ancestral order `1, ..., p`, where `1` has no parents.

set_cond_number *Set the condition number of the matrices in a sample of covariance/correlation matrices*

Description

Set the condition number of the matrices in a sample of covariance/correlation matrices

Usage

```
set_cond_number(sample, k)
```

Arguments

`sample`	Array, the `p x p x N` matrix sample
`k`	Condition number to be set

Value

A `p x p x N` array containing the matrices with the fixed condition number

`ug-constrained covariance matrices`

Simulation of covariance matrices.

Description

Sample covariance matrices, possibly with a zero pattern constrained by an undirected graph.

Usage

```
port(N = 1, p = 3, d = 1, ug = NULL, zapzeros = TRUE)

diagdom(N = 1, p = 3, d = 1, ug = NULL)
```

Arguments

`N`	Number of samples.
`p`	Matrix dimension. Ignored if ug is provided.
`d`	Number in `[0,1]`, the proportion of non-zero entries in the sampled matrices. Ignored if ug is provided.
`ug`	An igraph undirected graph specifying the zero pattern in the sampled matrices.
`zapzeros`	Boolean, convert to zero extremely low entries? Defaults to `TRUE`.

Details

Function `port()` uses the method described in Córdoba et al. (2018). In summary, it consists on generating a random matrix `Q` and performing row-wise orthogonalization such that if `i` and `j` are not adjacent in ug, then the rows corresponding to such indices are orthogonalized, without violating previous orthogonalizations and without introducing unwanted independences. The resulting matrix after the process has finished is the cross product of `Q`.

We also provide an implementation of the most commonly used in the literature `diagdom()`. By contrast, this method produces a random matrix `M` with zeros corresponding to missing edges in ug, and then enforces a dominant diagonal to ensure positive definiteness. Matrices produced by `diagdom` usually are better conditioned than those by `port`; however, they typically suffer from small off-diagonal entries, which can compromise model validation in Gaussian graphical models. This is avoided by `port`.

Value

A three-dimensional array of length `p x p x N`

References

Córdoba, I., Varando, G., Bielza, C. and Larrañaga, P. A partial orthogonalization method for simulation covariance and concentration graph matrices. *Proceedings of Machine Learning Research* (PGM 2018), vol. 72, pp. 61 - 72, 2018.

Examples

```
## Partial orthogonalization
# Generate a full matrix (default behaviour)
port()

# Generate a matrix with a percentage of zeros
port(d = 0.5)
port(d = 0.5, zapzeros = FALSE) # no zero zap

# Generate a random undirected graph structure
ug <- rgraph(p = 3, d = 0.5)
igraph::print.igraph(ug)

# Generate a matrix complying with the predefined zero pattern
port(ug = ug)
port(ug = ug, zapzeros = FALSE) # no zero zap
## Diagonal dominance
# Generate a full matrix (default behaviour)
diagdom()

# Generate a matrix with a percentage of zeros
diagdom(d = 0.5)

# Generate a matrix complying with the predefined zero pattern
igraph::print.igraph(ug)
diagdom(ug = ug)
```

`vectorize` *Vectorize a sample of covariance/correlation matrices*

Description

Vectorize a sample of covariance/correlation matrices

Usage

```
vectorize(sample)
```

Arguments

`sample`	Array, the `p x p x N` sample to vectorize

Details

Note that if the sample is of covariance matrices, as returned by `port()` and `diagdom()`, the diagonal is omitted from the vectorization process.

Value

A `p*(p - 1)/2 x N` matrix containing the vectorized sample

Index

anti_t, 2

chol_iid *(*dag-constrained correlation matrices*)*, 2
chol_iid(), *3*
chol_mh *(*dag-constrained correlation matrices*)*, 2
chol_mh(), *3*
chol_polar *(*dag-constrained correlation matrices*)*, 2
chol_polar(), *3*

dag-constrained correlation matrices, 2
diagdom *(*ug-constrained covariance matrices*)*, 7
diagdom(), *7*, *8*

gmat, 4
gmat-package *(*gmat*)*, 4

metropolis-hastings sampling, 4
mh_sphere *(*metropolis-hastings sampling*)*, 4
mh_sphere(), *5*
mh_u *(*metropolis-hastings sampling*)*, 4
mh_u(), *3*, *5*

port *(*ug-constrained covariance matrices*)*, 7
port(), *7*, *8*

rgraph, 6

set_cond_number, 6

ug-constrained covariance matrices, 7

vectorize, 8