Bayesian Structure Learning in Graphical Models

Description

The R package **BDgraph** provides statistical tools for Bayesian structure learning in undirected graphical models for continuous, ordinal/count/discrete, binary, and mixed data. The package is implemented the recent improvements in the Bayesian graphical models’ literature, including Mohammadi and Wit (2015), Mohammadi et al. (2021), Mohammadi et al. (2017), and Dobra and
Mohammadi (2018). To speed up the computations, the intensive tasks of the package are implemented in parallel using OpenMP in C++ and interfaced with R. Besides, the package contains several functions for simulation and visualization, as well as several multivariate datasets taken from the literature.

How to cite this package

To cite BDgraph in publications use:


Author(s)

Reza Mohammadi [aut, cre] (https://orcid.org/0000-0001-9538-0648),
Ernst Wit [aut] (https://orcid.org/0000-0002-3671-9610),
Adrian Dobra [ctb] (https://orcid.org/0000-0001-7793-2197).
Maintainer: Reza Mohammadi <a.mohammadi@uva.nl>

References

adj2link

Extract links from an adjacency matrix

Description

Extract links from an adjacency matrix or an object of classes "sim" from function `bdgraph.sim` and "graph" from function `graph.sim`.

See Also

`bdgraph`, `bdgraph.mpl`, `bdgraph.dw`, `bdgraph.sim`, `compare`, `rgwish`

Examples

```r
## Not run:
library( BDgraph )

set.seed( 10 )

# Generating multivariate normal data from a 'scale-free' graph
data.sim <- bdgraph.sim( n = 100, p = 10, graph = "scale-free", vis = TRUE )

# Running algorithm based on GGMs
bdgraph.obj <- bdgraph( data = data.sim, iter = 5000 )
summary( bdgraph.obj )

# To compare the result with true graph
compare( bdgraph.obj, data.sim, main = c( "Target", "BDgraph" ), vis = TRUE )

# Confusion Matrix
conf.mat( actual = data.sim, pred = bdgraph.obj )
conf.mat.plot( actual = data.sim, pred = bdgraph.obj )

# Running algorithm based on GGMs and marginal pseudo-likelihood
bdgraph.mpl.obj <- bdgraph.mpl( data = data.sim, iter = 5000 )
summary( bdgraph.mpl.obj )

# Confusion Matrix
conf.mat( actual = data.sim, pred = bdgraph.mpl.obj )
conf.mat.plot( actual = data.sim, pred = bdgraph.mpl.obj )

# To compare the results of both algorithms with true graph
compare( list( bdgraph.obj, bdgraph.mpl.obj ), data.sim,
         main = c( "Target", "BDgraph", "BDgraph_mpl" ), vis = TRUE )
```

## End(Not run)

adj2link

Extract links from an adjacency matrix
adj2link

Usage

adj2link(adj)

Arguments

adj adjacency matrix corresponding to a graph structure in which $a_{ij} = 1$ if there is a link between notes $i$ and $j$, otherwise $a_{ij} = 0$. It can be an object with S3 class "sim" from function bdgraph.sim. It can be an object with S3 class "graph" from function graph.sim.

Value

matrix corresponding to the extracted links from graph structure.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>

References


See Also

link2adj, graph.sim

Examples

# Generating a 'random' graph
adj <- graph.sim( p = 6, vis = TRUE )

adj2link(adj)
Compute the area under the ROC curve

Description

This function computes the numeric value of area under the ROC curve (AUC) specifically for graph structure learning.

Usage

```r
auc( pred, actual, cut = 200, calibrate = TRUE )
```

Arguments

- `pred`: adjacency matrix corresponding to an estimated graph. It can be an object with S3 class "bdgraph" from function `bdgraph`. It can be an object of S3 class "ssgraph", from the function `ssgraph::ssgraph()` of R package `ssgraph::ssgraph()`. It can be a numeric or ordered vector of the same length than `actual`, containing the predicted value of each observation.

- `actual`: adjacency matrix corresponding to the actual graph structure in which $a_{ij} = 1$ if there is a link between notes $i$ and $j$, otherwise $a_{ij} = 0$. It can be an object with S3 class "sim" from function `bdgraph.sim`. It can be an object with S3 class "graph" from function `graph.sim`. It can be a factor, numeric or character vector of responses (true class), typically encoded with 0 (controls) and 1 (cases). Only two classes can be used in a ROC curve.

- `cut`: number of cut points.

- `calibrate`: If TRUE, compute the value of AUC by taking the level of the probabilities into account.

Value

The numeric AUC value

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>; Lucas Vogels <l.f.o.vogels@uva.nl>

References

bdgraph

See Also

plotroc, pROC::plot.roc(), pROC::auc(), pROC::print.roc(), bdgraph, bdgraph.mpl, compare

Examples

## Not run:
set.seed( 5 )

# Generating multivariate normal data from a 'scale-free' graph
data.sim = bdgraph.sim( n = 200, p = 15, graph = "scale-free", vis = TRUE )

# Running BDMCMC algorithm
sample.bdmcmc = bdgraph( data = data.sim, algorithm = "bdmcmc", iter = 10000 )
BDgraph::auc( pred = sample.bdmcmc, actual = data.sim, calibrate = TRUE )

## End(Not run)

bdgraph

Search algorithm in graphical models

Description

As the main function of the BDgraph package, this function consists of several MCMC sampling algorithms for Bayesian model determination in undirected graphical models. To speed up the computations, the birth-death MCMC sampling algorithms are implemented in parallel using OpenMP in C++.

Usage

bdgraph( data, n = NULL, method = "ggm", algorithm = "bdmcmc", iter = 5000,
burnin = iter / 2, not.cont = NULL, g.prior = 0.2, df.prior = 3,
g.start = "empty", jump = NULL, save = FALSE,
cores = NULL, threshold = 1e-8, verbose = TRUE, nu = 1 )

Arguments

data there are two options: (1) an \((n \times p)\) matrix or a data.frame corresponding to the data. (2) an \((p \times p)\) covariance matrix as \(S = X'X\) which \(X\) is the data matrix \((n\) is the sample size and \(p\) is the number of variables). It also could be an object of class "sim", from function bdgraph.sim. The input matrix is automatically identified by checking the symmetry.
number of observations. It is needed if the "data" is a covariance matrix.
method character with two options "ggm" (default) and "gcgm". Option "ggm" is for Gaussian graphical models based on Gaussianity assumption. Option "gcgm" is for Gaussian copula graphical models for the data that do not follow Gaussianity assumption (e.g. continuous non-Gaussian, count, or mixed dataset).
algorithm: character with two options "bdmcmc" (default) and "rjmcmc". Option "bdmcmc" is based on birth-death MCMC algorithm. Option "rjmcmc" is based on reversible jump MCMC algorithm. Option "bd-dmh" is based on birth-death MCMC algorithm using double Metropolis Hasting. Option "rj-dmh" is based on reversible jump MCMC algorithm using double Metropolis Hasting.

iter: number of iteration for the sampling algorithm.

burnin: number of burn-in iteration for the sampling algorithm.

not.cont: for the case method = "gcgm", a vector with binary values in which 1 indicates not continuous variables.

g.prior: for determining the prior distribution of each edge in the graph. There are two options: a single value between 0 and 1 (e.g. 0.5 as a noninformative prior) or an \((p \times p)\) matrix with elements between 0 and 1.

df.prior: degree of freedom for G-Wishart distribution, \(W_{G}(b, D)\), which is a prior distribution of the precision matrix.

g.start: corresponds to a starting point of the graph. It could be an \((p \times p)\) matrix, "empty" (default), or "full". Option "empty" means the initial graph is an empty graph and "full" means a full graph. It also could be an object with S3 class "bdgraph" of R package BDgraph or the class "ssgraph" of R package ssgraph::ssgraph(); this option can be used to run the sampling algorithm from the last objects of previous run (see examples).

jump: it is only for the BDMCMC algorithm (algorithm = "bdmcmc"). It is for simultaneously updating multiple links at the same time to update graph in the BDMCMC algorithm.

save: logical: if FALSE (default), the adjacency matrices are NOT saved. If TRUE, the adjacency matrices after burn-in are saved.

cores: number of cores to use for parallel execution. The case cores = "all" means all CPU cores to use for parallel execution.

threshold: threshold value for the convergence of sampling algorithm from G-Wishart for the precision matrix.

verbose: logical: if TRUE (default), report/print the MCMC running time.

nu: prior parameter for option method = "tgm".

Value

An object with S3 class "bdgraph" is returned:

p_links: upper triangular matrix which corresponds the estimated posterior probabilities of all possible links.

K_hat: posterior estimation of the precision matrix.

For the case "save = TRUE" is returned:

sample_graphs: vector of strings which includes the adjacency matrices of visited graphs after burn-in.

graph_weights: vector which includes the waiting times of visited graphs after burn-in.
all_graphs vector which includes the identity of the adjacency matrices for all iterations after burn-in. It is needed for monitoring the convergence of the BD-MCMC algorithm.

all_weights vector which includes the waiting times for all iterations after burn-in. It is needed for monitoring the convergence of the BD-MCMC algorithm.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl> and Ernst Wit

References


See Also

bdgraph.mpl, bdgraph.dw, bdgraph.sim, summary.bdgraph, compare

Examples

```r
## Not run:
set.seed( 10 )

# - - Example 1

# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 100, p = 10, size = 15, vis = TRUE )

bdgraph.obj <- bdgraph( data = data.sim, iter = 1000, save = TRUE )

summary( bdgraph.obj )

# Confusion Matrix
conf.mat( actual = data.sim, pred = bdgraph.obj )

conf.mat.plot( actual = data.sim, pred = bdgraph.obj )
```
# To compare our result with true graph

```r
compare( bdgraph.obj, data.sim, main = c( "Target", "BDgraph" ), vis = T )
```

# Running algorithm with starting points from previous run

```r
bdgraph.obj2 <- bdgraph( data = data.sim, g.start = bdgraph.obj )
```

```r
compare( list( bdgraph.obj, bdgraph.obj2 ), data.sim,
  main = c( "Target", "Frist run", "Second run" ) )
```

# -- Example 2

# Generating mixed data from a 'scale-free' graph

```r
data.sim <- bdgraph.sim( n = 200, p = 7, type = "mixed", graph = "scale-free", vis = TRUE )
```

```r
bdgraph.obj <- bdgraph( data = data.sim, method = "gcgm" )
```

```r
summary( bdgraph.obj )
```

```r
compare( bdgraph.obj, data.sim, vis = T )
```

```r
conf.mat( actual = data.sim, pred = bdgraph.obj )
```

```r
conf.mat.plot( actual = data.sim, pred = bdgraph.obj )
```

## End(Not run)

---

**bdgraph.dw**

Search algorithm for Gaussian copula graphical models for count data

**Description**

This function consists of several sampling algorithms for Bayesian structure learning in undirected graphical models for count data. It is based on Gaussian copula graphical models with discrete Weibull distributed marginals. To speed up the computations, the birth-death MCMC sampling algorithms are implemented in parallel using **OpenMP** in C++.

**Usage**

```r
bdgraph.dw( data, x = NULL, formula = y ~ .,
  n = NULL, algorithm = "bdmcmc", iter = 5000,
  burnin = iter / 2, g.prior = 0.2, df.prior = 3,
  ZI = FALSE, iter_bdw = 5000,
  g.start = "empty", jump = NULL, save = FALSE,
  q = NULL, beta = NULL, pii = NULL,
  cores = NULL, threshold = 1e-8, verbose = TRUE )
```
Arguments

data \((n \times p)\) matrix or a \texttt{data.frame} corresponding to the data on the \(p\) nodes of the graph. It can also be an object of class "sim", from the function \texttt{bdgraph.sim}.

\(x\) \((n \times k)\) matrix or a \texttt{data.frame} corresponding to the predictors.

\texttt{formula}\ object of class \texttt{formula} as a symbolic description of the model for linking each node to the predictors. For the case of \texttt{data.frame}, it is taken as the model frame (see \texttt{model.frame}).

\(n\) number of observations. It is needed if the "data" is a covariance matrix.

\texttt{algorithm}\ character with two options "bdmcmc" (default) and "rjmcmc". Option "bdmcmc" is based on a birth-death MCMC algorithm. Option "rjmcmc" is based on a reversible jump MCMC algorithm.

\texttt{iter}\ number of iterations for the sampling algorithm for graph learning.

\texttt{burnin}\ number of burn-in iterations for the sampling algorithm for graph learning.

\texttt{g.prior}\ for determining the prior distribution of each edge in the graph. There are two options: a single value between 0 and 1 (e.g. 0.5 as a noninformative prior) or a \((p \times p)\) matrix with elements between 0 and 1.

\texttt{df.prior}\ degree of freedom for G-Wishart distribution, \(W_G(b, D)\), which is a prior distribution for the precision matrix.

\texttt{ZI}\ logical. If FALSE (default), the conditional distribution of each response variable is assumed to be Discrete Weibull given the predictors \(x\). If TRUE, a zero-inflated model will be applied to each response. \texttt{ZI} can be passed also as a vector, in order to specify which of the \(p\) variables should be fitted with zero-inflation (TRUE) or not (FALSE).

\texttt{iter_bdw}\ number of iterations for the sampling algorithm to estimate the regression parameters for the Discrete Weibull distribution. It is passed to the \texttt{bdw.reg} function.

\texttt{g.start}\ corresponds to a starting point of the graph. It could be an \((p \times p)\) matrix, "empty" (default), or "full". Option "empty" means that the initial graph is an empty graph and "full" means a full graph. It also could be an object with \(S3\) class "bdgraph" of \texttt{R} package \texttt{BDgraph} or the class "ssgraph" of \texttt{R} package \texttt{ssgraph::ssgraph()}; this option can be used to run the sampling algorithm from the last objects of the previous run (see examples).

\texttt{jump}\ it is only for the BDMCMC algorithm (algorithm = "bdmcmc"). It is for simultaneously updating multiple links at the same time while updating the graph in the BDMCMC algorithm.

\texttt{save}\ logical: if FALSE (default), the adjacency matrices are NOT saved. If TRUE, the adjacency matrices after burn-in are saved.

\texttt{q, beta}\ parameters of the discrete Weibull distribution used for the marginals. They should be given either as a \((n \times p)\) matrix (if covariates are present) or as a vector (if covariates are not present). If NULL (default), these parameters are estimated by the \texttt{bdw.reg} function.

\texttt{pii}\ vector of zero-inflation parameters of the zero-inflated discrete Weibull distributions used for the marginals. If NULL (default), this parameter is estimated by the \texttt{bdw.reg} function when \texttt{ZI} = TRUE.
cores  number of cores to use for parallel execution. The case cores = "all" means all CPU cores to use for parallel execution.

threshold  threshold value for the convergence of the sampling algorithm from G-Wishart for the precision matrix.

verbose  logical: if TRUE (default), report/print the MCMC running time.

Value

An object with S3 class "bdgraph" is returned, containing:

p_links  upper triangular matrix corresponding to the estimated posterior probabilities of all possible links.

K_hat  posterior estimation of the precision matrix.

sample_marginals  posterior samples of the regression coefficients of the marginal distributions.

For the case "save = TRUE", the code returns:

sample_graphs  vector of strings which includes the adjacency matrices of the graphs visited after burn-in.

graph_weights  vector which includes the waiting times of the graphs visited after burn-in.

all_graphs  vector which includes the identity of the adjacency matrices for all iterations after burn-in. It is needed for monitoring the convergence of the BDMCMC algorithm.

all_weights  vector which includes the waiting times for all iterations after burn-in. It is needed for monitoring the convergence of the BDMCMC algorithm.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>, Veronica Vinciotti, and Pariya Behrouzi

References


See Also

bdgraph, bdgraph.mpl, bdw.reg, bdgraph.sim, summary.bdgraph, compare

Examples

```r
## Not run:
# - - Example 1
# Generating multivariate Discrete Weibull data based on 'random' graph
data.sim <- bdgraph.sim( n = 100, p = 10, type = "dw", vis = TRUE )

bdgraph.obj <- bdgraph.dw( data = data.sim, iter = 5000 )

summary( bdgraph.obj )

# To compare the result with true graph
compare( bdgraph.obj, data.sim, main = c("Target", "BDgraph"), vis = TRUE )

# - - Example 2
# Generating multivariate Discrete Weibull data based on a 'scale-free' graph
data.sim <- bdgraph.sim( n = 100, p = 10, type = "dw", graph = "scale-free", vis = TRUE )

bdgraph.obj <- bdgraph.dw( data = data.sim, iter = 10000 )

summary( bdgraph.obj )

compare( bdgraph.obj, data.sim, main = c("Target", "BDgraph"), vis = TRUE )

## End(Not run)
```

bdgraph.mpl

Search algorithm in graphical models using marginal pseudo-likelihood

Description

This function consists of several sampling algorithms for Bayesian model determination in undirected graphical models based on marginal pseudo-likelihood. To speed up the computations, the birth-death MCMC sampling algorithms are implemented in parallel using OpenMP in C++. 

---


See Also

bdgraph, bdgraph.mpl, bdw.reg, bdgraph.sim, summary.bdgraph, compare

Examples

```r
## Not run:
# - - Example 1
# Generating multivariate Discrete Weibull data based on 'random' graph
data.sim <- bdgraph.sim( n = 100, p = 10, type = "dw", vis = TRUE )

bdgraph.obj <- bdgraph.dw( data = data.sim, iter = 5000 )

summary( bdgraph.obj )

# To compare the result with true graph
compare( bdgraph.obj, data.sim, main = c("Target", "BDgraph"), vis = TRUE )

# - - Example 2
# Generating multivariate Discrete Weibull data based on a 'scale-free' graph
data.sim <- bdgraph.sim( n = 100, p = 10, type = "dw", graph = "scale-free", vis = TRUE )

bdgraph.obj <- bdgraph.dw( data = data.sim, iter = 10000 )

summary( bdgraph.obj )

compare( bdgraph.obj, data.sim, main = c("Target", "BDgraph"), vis = TRUE )

## End(Not run)
```
bdgraph.mpl

Usage

bdgraph.mpl( data, n = NULL, method = "ggm", transfer = TRUE,
  algorithm = "bdmcmc", iter = 5000, burnin = iter / 2,
  g.prior = 0.2, g.start = "empty",
  jump = NULL, alpha = 0.5, save = FALSE,
  cores = NULL, operator = "or", verbose = TRUE )

Arguments

data there are two options: (1) an \((n \times p)\) matrix or a data.frame corresponding
to the data, (2) an \((p \times p)\) covariance matrix as \(S = X'X\) which \(X\) is the data
matrix \((n\) is the sample size and \(p\) is the number of variables). It also could
be an object of class "sim", from function bdgraph.sim. The input matrix is
automatically identified by checking the symmetry.

n number of observations. It is needed if the "data" is a covariance matrix.

method character with two options "ggm" (default), "dgm" and "dgm-binary". Option
"ggm" is for Gaussian graphical models based on Gaussianity assumption. Op-
tion "dgm" is for discrete graphical models for the count data. Option "dgm-binary"
is for discrete graphical models for the data that are binary.

transfer for only 'count' data which method = "dgm" or method = "dgm-binary".

algorithm character with two options "bdmcmc" (default) and "rjmcmc". Option "bdmcmc"
is based on birth-death MCMC algorithm. Option "rjmcmc" is based on reveri-
ble jump MCMC algorithm. Option "hc" is based on hill-climbing algorithm;
this algorithm is only for count data which method = "dgm" or method = "dgm-binary".

iter number of iteration for the sampling algorithm.

burnin number of burn-in iteration for the sampling algorithm.

g.prior for determining the prior distribution of each edge in the graph. There are two
options: a single value between 0 and 1 (e.g. 0.5 as a noninformative prior) or
an \((p \times p)\) matrix with elements between 0 and 1.

g.start corresponds to a starting point of the graph. It could be an \((p \times p)\) matrix,
"empty" (default), or "full". Option "empty" means the initial graph is an
empty graph and "full" means a full graph. It also could be an object with
S3 class "bdgraph" of R package BDgraph or the class "ssgraph" of R package
ssgraph::ssgraph(); this option can be used to run the sampling algorithm
from the last objects of previous run (see examples).

jump it is only for the BDMCMC algorithm (algorithm = "bdmcmc"). It is for si-
multaneously updating multiple links at the same time to update graph in the
BDMCMC algorithm.

alpha value of the hyper parameter of Dirichlet, which is a prior distribution.

save logical: if FALSE (default), the adjacency matrices are NOT saved. If TRUE,
the adjacency matrices after burn-in are saved.

cores number of cores to use for parallel execution. The case cores = "all" means
all CPU cores to use for parallel execution.

operator character with two options "or" (default) and "and". It is for hill-climbing algo-

verbose logical: if TRUE (default), report/print the MCMC running time.
Value

An object with S3 class "bdgraph" is returned:

- **p_links**: upper triangular matrix which corresponds the estimated posterior probabilities of all possible links.

For the case "save = TRUE" is returned:

- **sample_graphs**: vector of strings which includes the adjacency matrices of visited graphs after burn-in.
- **graph_weights**: vector which includes the waiting times of visited graphs after burn-in.
- **all_graphs**: vector which includes the identity of the adjacency matrices for all iterations after burn-in. It is needed for monitoring the convergence of the BD-MCMC algorithm.
- **all_weights**: vector which includes the waiting times for all iterations after burn-in. It is needed for monitoring the convergence of the BD-MCMC algorithm.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>, Adrian Dobra, and Johan Pensar

References


See Also

bdgraph, bdgraph.dw, bdgraph.sim, summary.bdgraph, compare

Examples

```r
# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 70, p = 5, size = 7, vis = TRUE )

bdgraph.obj <- bdgraph.mpl( data = data.sim, iter = 500 )

summary( bdgraph.obj )

# To compare the result with true graph
compare( bdgraph.obj, data.sim, main = c( "Target", "BDgraph" ) )
```
bdgraph.npn

Nonparametric transfer

Description

Transfers non-Gaussian data to Gaussian.

Usage

bdgraph.npn( data, npn = "shrinkage", npn.thresh = NULL )

Arguments

data: \((n \times p)\) matrix or a data.frame corresponding to the data \((n\) is the sample size and \(p\) is the number of variables).

npn: character with three options "shrinkage" (default), "truncation", and "skeptic". Option "shrinkage" is for the shrunken transformation, option "truncation" is for the truncated transformation and option "skeptic" is for the non-paranormal skeptic transformation. For more details see references.

npn.thresh: truncation threshold; it is only for the truncated transformation \((npn= "truncation")\). The default value is \(1/(4n^{1/4}\sqrt{\pi\log(n)})\).

Value

\((n \times p)\) matrix of transferred data, if \(npn = "shrinkage"\) or "truncation", and a non-paranormal correlation \((p \times p)\) matrix, if \(npn = "skeptic"\).

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>

References


See Also

bdgraph.sim, bdgraph, bdgraph.mpl
bdgraph.sim

Examples

```
## Not run:
# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 6, p = 4, size = 4 )
data <- ( data.sim $ data - 3 ) ^ 4
data

data # Transfer the data by truncation
bdgraph.npn( data, npn = "truncation" )

data # Transfer the data by shrunken
bdgraph.npn( data, npn = "shrunken" )

data # Transfer the data by skeptic
bdgraph.npn( data, npn = "skeptic" )

## End(Not run)
```

bdgraph.sim

Graph data simulation

Description

Simulating multivariate distributions with different types of underlying graph structures, including "random", "cluster", "smallworld", "scale-free", "lattice", "hub", "star", "circle", "AR(1)", and "AR(2)". Based on the underlying graph structure, the function generates different types of multivariate data, including "Gaussian", "non-Gaussian", "categorical", "pois" (Poisson), "nbinom" (negative binomial), "dweibull" (discrete Weibull), "binary", "t" (t-distribution), "alternative-t", or "mixed" data. This function can be used also for simulating only graphs by setting the option n=0 (default).

Usage

```
bdgraph.sim( p = 10, graph = "random", n = 0, type = "Gaussian", prob = 0.2,
size = NULL, mean = 0, class = NULL, cut = 4, b = 3,
D = diag( p ), K = NULL, sigma = NULL,
q = exp(-1), beta = 1, vis = FALSE, rewire = 0.05,
range.mu = c( 3, 5 ), range.dispersion = c( 0.01, 0.1 ), nu = 1 )
```

Arguments

- `p`: number of variables (nodes).
- `graph`: graph structure with options "random", "cluster", "smallworld", "scale-free", "lattice", "hub", "star", "circle", "AR(1)", and "AR(2)". It could also be an adjacency matrix corresponding to a graph structure (an upper triangular matrix in which \( g_{ij} = 1 \) if there is a link between nodes \( i \) and \( j \), otherwise \( g_{ij} = 0 \)).
n number of samples required. Note that for the case \( n = 0 \), only the graph is generated.

type type of data with options "Gaussian" (default), "non-Gaussian", "categorical", "pois", "nbinom", "dweibull", "binary", "mixed", "t", and "alternative-t". For the option "Gaussian", data are generated from a multivariate normal distribution. For the option "non-Gaussian", data are transferred from a multivariate normal distribution to a continuous multivariate non-Gaussian distribution via Exponential marginals. For the option "categorical", data are transferred from a multivariate normal distribution to multivariate 'categorical' data. For the option "pois", data are transferred from a multivariate normal distribution to a multivariate Poisson distribution. For the option "nbinom", data are transferred from a multivariate normal distribution to a multivariate Negative Binomial distribution. For the option "dweibull", data are transferred from a multivariate normal distribution to a multivariate discrete Weibull distribution with parameters \( q \) and \( \beta \). For the option "binary", data are generated directly from the joint distribution, in this case \( p \) must be less than 17. For the option "mixed", data are transferred from a multivariate normal distribution to a mixture of 'categorical', 'non-Gaussian', 'binary' and 'Gaussian', respectively.

prob if graph = "random", it is the probability that a pair of nodes has a link.

size number of links in the true graph (graph size).

mean vector specifying the mean of the variables.

class if graph = "cluster", it is the number of classes.

cut if type = "categorical", it is the number of categories for simulating 'categorical' data.

b degree of freedom for G-Wishart distribution, \( W_G(b, D) \).

D positive definite \((p \times p)\) "scale" matrix for G-Wishart distribution, \( W_G(b, D) \). The default is an identity matrix.

K if graph = "fixed", it is a positive-definite symmetric matrix, corresponding to the true precision matrix.

sigma if graph = "fixed", it is a positive-definite symmetric matrix corresponding to the true covariance matrix.

q, beta if type = "dweibull", they are the parameters of the discrete Weibull distribution with density

\[
p(x, q, \beta) = q^x \beta - q^{(x+1)} \beta, \quad \forall x = \{0, 1, 2, \ldots\}.
\]

They can be given either as a vector of length \( p \) or as an \((n \times p)\) matrix, e.g. if covariates are available and a regression model is used.

vis visualize the true graph structure.

rewire rewiring probability for smallworld network. Must be between 0 and 1.

range.mu, range.dispersion if type = "nbinom", vector with two elements specifying the range of parameters for the Negative Binomial distribution.

nu if type = "t" or "alternative-t", it is the parameter of the t distribution with density.
Value

An object with S3 class "sim" is returned:

- **data**: generated data as an $(n \times p)$ matrix.
- **sigma**: covariance matrix of the generated data.
- **K**: precision matrix of the generated data.
- **G**: adjacency matrix corresponding to the true graph structure.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>, Pariya Behrouzi, Veronica Vinciotti, Ernst Wit, and Alexander Christensen

References


See Also

- `graph.sim`, `bdgraph`, `bdgraph.mpl`

Examples

```r
## Not run:
# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( p = 10, n = 50, prob = 0.3, vis = TRUE )
print( data.sim )

# Generating multivariate normal data from a 'hub' graph
data.sim <- bdgraph.sim( p = 6, n = 3, graph = "hub", vis = FALSE )
round( data.sim $ data, 2 )

# Generating mixed data from a 'hub' graph
data.sim <- bdgraph.sim( p = 8, n = 10, graph = "hub", type = "mixed" )
round( data.sim $ data, 2 )

# Generating only a 'scale-free' graph (with no data)
graph.sim <- bdgraph.sim( p = 8, graph = "scale-free" )
plot( graph.sim )
graph.sim $ G
## End(Not run)
```
Bayesian estimation of (zero-inflated) Discrete Weibull regression

Description

Bayesian estimation of the parameters for Discrete Weibull (DW) regression. The conditional distribution of the response given the predictors is assumed to be DW with parameters q and beta, dependent on the predictors, and, with an additional parameter pi under zero inflation.

Usage

```r
bdw.reg( data, formula = NA, iter = 5000, burnin = NULL,
    dist.q = dnorm, dist.beta = dnorm,
    par.q = c( 0, 1 ), par.beta = c( 0, 1 ), par.pi = c( 1, 1 ),
    initial.q = NULL, initial.beta = NULL, initial.pi = NULL,
    ZI = FALSE, scale.proposal = NULL, adapt = TRUE, print = TRUE )
```

Arguments

data data.frame or matrix corresponding to the data, containing the variables in the model.

formula object of class formula as a symbolic description of the model to be fitted. For the case of data.frame, it is taken as the model frame (see model.frame).

iter number of iterations for the sampling algorithm.

burnin number of burn-in iterations for the sampling algorithm.

dist.q Prior density for the regression coefficients associated to the parameter q. The default is a Normal distribution (dnorm). Any density function which has two parameters and can support the log = TRUE flag can be used, e.g. dnorm, dlnorm, dunif etc.

dist.beta Prior density for the regression coefficients associated to the parameter beta. The default is a Normal distribution (dnorm). Any density function which has two parameters and can support the log = TRUE flag can be used, e.g. dnorm, dlnorm, dunif etc.

par.q vector of length two corresponding to the parameters of dist.q.

par.beta vector of length two corresponding to the parameters of dist.beta.

par.pi vector of length two corresponding to the parameters of the beta prior density on pi.

initial.q, initial.beta, initial.pi vector of initial values for the regression coefficients and for pi (if ZI = TRUE).

ZI logical: if FALSE (default), the conditional distribution of the response given the predictors is assumed to be DW with parameters q and beta. If TRUE, a zero-inflated DW distribution will be applied.

scale.proposal scale of the proposal function. Setting to lower values results in an increase in the acceptance rate of the sampler.
adapt logical: if TRUE (default), the proposals will be adapted. If FALSE, no adapting will be applied.

print logical: if TRUE (default), tracing information is printed.

Details

The regression model uses a logit link function on \( q \) and a log link function on \( \beta \), the two parameters of a DW distribution, with probability mass function given by

\[
DW(y) = q^y - q^{(y+1)}, \quad y = 0, 1, 2, \ldots
\]

For the case of zero inflation (\( ZI = \text{TRUE} \)), a zero-inflated DW is considered:

\[
f(y) = (1 - pi)I(y = 0) + pi DW(y)
\]

where \( 0 \leq pi \leq 1 \) and \( I(y = 0) \) is an indicator for the point mass at zero for the response \( y \).

Value

- sample: MCMC samples
- q.est: posterior estimates of \( q \)
- beta.est: posterior estimates of \( \beta \)
- pi.est: posterior estimates of \( pi \)
- accept.rate: acceptance rate of the MCMC algorithm

Author(s)

Veronica Vinciotti, Reza Mohammadi <a.mohammadi@uva.nl>, and Pariya Behrouzi

References


See Also

bdgraph.dw, bdgraph, ddweibull, bdgraph.sim
Examples

## Not run:
# - - Example 1

q = 0.6
beta = 1.1
n = 500

y = BDgraph::rdweibull( n=n, q=q, beta=beta )

output = bdw.reg( data = y, y ~ ., iter = 5000 )

output $ q.est
output $ beta.est

traceplot( output $ sample[, 1], acf = T, pacf = T )
traceplot( output $ sample[, 2], acf = T, pacf = T )

# - - Example 2

q = 0.6
beta = 1.1
pii = 0.8
n = 500

y_dw = BDgraph::rdweibull( n=n, q=q, beta=beta )
z = rbinom( n=n, size = 1, prob = pii )
y = z * y_dw

output = bdw.reg( data = y, iter = 5000, ZI = TRUE )

output $ q.est
output $ beta.est
output $ pi.est

traceplot( output $ sample[, 1], acf = T, pacf = T )
traceplot( output $ sample[, 2], acf = T, pacf = T )
traceplot( output $ sample[, 3], acf = T, pacf = T )

# - - Example 3

theta.q = c( 0.1, -0.1, 0.34 ) # true parameter
theta.beta = c( 0.1, -.15, 0.5 ) # true parameter

n = 500

x1 = runif( n=n, min = 0, max = 1.5 )
x2 = runif( n=n, min = 0, max = 1.5 )

q = 1 / ( 1 + exp( -reg_q ) )
beta = exp( reg_beta )
y = BDgraph::rdweibull( n = n, q = q, beta = beta )
data = data.frame( x1, x2, y )

output = bdw.reg( data, y ~ ., iter = 5000 )

# - - Example 4
theta.q = c( 1, -1, 0.8 ) # true parameter
theta.beta = c( 1, -1, 0.3 ) # true parameter
pii = 0.8

n = 500
x1 = runif( n = n, min = 0, max = 1.5 )
x2 = runif( n = n, min = 0, max = 1.5 )

q = 1 / ( 1 + exp( - reg_q ) )

beta = exp( reg_beta )
y_dw = BDgraph::rdweibull( n = n, q = q, beta = beta )
z = rbinom( n = n, size = 1, prob = pii )
y = z * y_dw

data = data.frame( x1, x2, y )

output = bdw.reg( data, y ~ ., iter = 5000 )

## End(Not run)

---

**bf**

Bayes factor for two graphs

**Description**

Compute the Bayes factor for two graph structures.

**Usage**

```r
bf( num, den, bdgraph.obj, log = TRUE )
```
Arguments

\texttt{num, den} \hspace{1cm} \text{adjacency matrix corresponding to the true graph structure in which } a_{ij} = 1 \text{ if there is a link between notes } i \text{ and } j, \text{ otherwise } a_{ij} = 0. \text{ It can be an object with S3 class } \text{"graph" from function } \text{\texttt{graph.sim}}. \text{ It can be an object with S3 class } \text{"sim" from function } \text{\texttt{bdgraph.sim}}.

\texttt{bdgraph.obj} \hspace{1cm} \text{object of S3 class } \text{"bdgraph" from function } \text{\texttt{bdgraph}}. \text{ It also can be an object of S3 class } \text{"ssgraph" from the function } \text{\texttt{ssgraph::ssgraph()}} \text{ of R package } \text{\texttt{ssgraph::ssgraph()}}.

\texttt{log} \hspace{1cm} \text{character value. If TRUE the Bayes factor is given as log(BF).}

Value

\text{single numeric value, the Bayes factor of the two graph structures } \text{num and den}.

Author(s)

Reza Mohammadi \texttt{<a.mohammadi@uva.nl>}

References


See Also

\texttt{bdgraph}, \texttt{bdgraph.mpl}, \texttt{compare.bdgraph.sim}

Examples

\texttt{## Not run:}
\texttt{
\# Generating multivariate normal data from a 'circle' graph}
\texttt{data.sim <- bdgraph.sim( n = 50, p = 6, graph = "circle", vis = TRUE )}

\texttt{
\# Running sampling algorithm}
\texttt{bdgraph.obj <- bdgraph( data = data.sim )}

\texttt{\# Running sampling algorithm}
\texttt{graph_1 <- graph.sim( p = 6, vis = TRUE )}

\texttt{\# Running sampling algorithm}
\texttt{graph_2 <- graph.sim( p = 6, vis = TRUE )}

\texttt{bf( num = graph_1, den = graph_2, bdgraph.obj = bdgraph.obj )}

\texttt{## End(Not run)}
**compare**

Graph structure comparison

**Description**

This function provides several measures to assess the performance of the graphical structure learning.

**Usage**

`compare( pred, actual, main = NULL, vis = FALSE )`

**Arguments**

- **pred**: adjacency matrix corresponding to an estimated graph. It can be an object with S3 class "bdgraph" from function `bdgraph`. It can be an object of S3 class "ssgraph", from the function `ssgraph::ssgraph()` of R package `ssgraph::ssgraph()`. It can be an object of S3 class "select", from the function `huge.select` of R package `huge`. It also can be a list of above objects for comparing two or more different approaches.

- **actual**: adjacency matrix corresponding to the true graph structure in which $a_{ij} = 1$ if there is a link between notes $i$ and $j$, otherwise $a_{ij} = 0$. It can be an object with S3 class "sim" from function `bdgraph.sim`. It can be an object with S3 class "graph" from function `graph.sim`. It can be an object with S3 class "graph" from function `graph.sim`.

- **main**: character vector giving the names for the result table.

- **vis**: logical: if TRUE, visualize the true graph and estimated graph structures.

**Value**

- **True positive**: number of correctly estimated links.
- **True negative**: number of true non-existing links which is correctly estimated.
- **False positive**: number of links which they are not in the true graph, but are incorrectly estimated.
- **False negative**: number of links which they are in the true graph, but are not estimated.
- **F1-score**: weighted average of the "positive predictive" and "true positive rate". The F1-score value reaches its best value at 1 and worst score at 0.
- **Specificity**: Specificity value reaches its best value at 1 and worst score at 0.
- **Sensitivity**: Sensitivity value reaches its best value at 1 and worst score at 0.
- **MCC**: Matthews Correlation Coefficients (MCC) value reaches its best value at 1 and worst score at 0.

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>, Antonio Abbruzzo, and Ivan Vujacic
References


See Also

bdgraph, bdgraph.mpl, bdgraph.sim, plotroc

Examples

```r
## Not run:
# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 50, p = 6, size = 7, vis = TRUE )

# Running sampling algorithm based on GGMs
sample.ggm <- bdgraph( data = data.sim, method = "ggm", iter = 10000 )

# Comparing the results
compare( sample.ggm, data.sim, main = c("True", "GGM"), vis = TRUE )

# Running sampling algorithm based on GCGMs
sample.gcgm <- bdgraph( data = data.sim, method = "gcgm", iter = 10000 )

# Comparing GGM and GCGM methods
compare( list( sample.ggm, sample.gcgm ), data.sim,
          main = c("True", "GGM", "GCGM"), vis = TRUE )

## End(Not run)
```

---

**conf.mat**  
*Confusion Matrix*

**Description**

Create a Confusion Matrix.

**Usage**

```r
conf.mat( pred, actual, cutoff = 0.5, proportion = FALSE,
          dnn = c("Prediction", "Actual"), ... )
```

**Arguments**

- **pred**: adjacency matrix corresponding to an estimated graph. It can be an object with S3 class "bdgraph" from function bdgraph. It can be an object of S3 class "ssgraph", from the function ssgraph::ssgraph() of R package ssgraph::ssgraph().
- **actual**: adjacency matrix representing the true graph.
- **cutoff**: threshold value used to determine the presence of an edge. Default is 0.5.
- **proportion**: logical value indicating whether to return the proportion of each category. Default is FALSE.
- **dnn**: character vector specifying the names of the prediction and actual categories. Default is c("Prediction", "Actual").
actual adjacency matrix corresponding to the actual graph structure in which $a_{ij} = 1$ if there is a link between notes $i$ and $j$, otherwise $a_{ij} = 0$. It can be an object with S3 class "sim" from function `bdgraph.sim`. It can be an object with S3 class "graph" from function `graph.sim`. It can be a factor, numeric or character vector of responses (true class), typically encoded with 0 (controls) and 1 (cases). Only two classes can be used in a ROC curve.

cutoff cutoff value for the case that pred is vector of probabilities. The default is 0.5.

proportion logical: FALSE (default) for a confusion matrix with number of cases. TRUE, for a confusion matrix with the proportion of cases.

dnn names to be given to the dimensions in the result (the dimnames names).

... further arguments to be passed to table.

Value the results of table on pred and actual.

Author(s) Reza Mohammadi <a.mohammadi@uva.nl>

See Also `conf.mat.plot`, `compare.roc`, `bdgraph`

Examples

```r
## Not run:
set.seed(100)

# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 50, p = 6, size = 7, vis = TRUE )

# Running sampling algorithm based on GGMs
sample.ggm <- bdgraph( data = data.sim, method = "ggm", iter = 10000 )

# Confusion Matrix for GGM method
conf.mat( pred = sample.ggm, actual = data.sim )

## End(Not run)
```

**conf.mat.plot**  
Plot Confusion Matrix

Description  
Plot a Confusion Matrix.
Usage

```r
conf.mat.plot( pred, actual, cutoff = 0.5, conf.level = 0, margin = 1,
       color = c( "#ff83a8", "#83ff9b" ), ... )
```

Arguments

- `pred`: adjacency matrix corresponding to an estimated graph. It can be an object with S3 class "bdgraph" from function `bdgraph`. It can be an object of S3 class "ssgraph", from the function `ssgraph::ssgraph()` of R package `ssgraph::ssgraph()`.
- `actual`: adjacency matrix corresponding to the actual graph structure in which $a_{ij} = 1$ if there is a link between notes $i$ and $j$, otherwise $a_{ij} = 0$. It can be an object with S3 class "sim" from function `bdgraph.sim`. It can be an object with S3 class "graph" from function `graph.sim`. It can be a factor, numeric or character vector of responses (true class), typically encoded with 0 (controls) and 1 (cases). Only two classes can be used in a ROC curve.
- `cutoff`: cutoff value for the case that pred is vector of probabilities. The default is 0.5.
- `conf.level`: confidence level used for the confidence rings on the odds ratios. Must be a single nonnegative number less than 1; if set to 0 (the default), confidence rings are suppressed.
- `margin`: numeric vector with the margins to equate. Must be one of 1 (the default), 2, or c(1, 2), which corresponds to standardizing the row, column, or both margins in each 2 by 2 table. Only used if std equals "margins".
- `color`: vector of length 2 specifying the colors to use for the smaller and larger diagonals of each 2 by 2 table.
- `...`: options to be passed to `fourfoldplot`.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>

See Also

`conf.mat`, `compare`, `roc`, `bdgraph`

Examples

```r
## Not run:
set.seed( 100 )

# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 50, p = 6, size = 7, vis = TRUE )

# Running sampling algorithm based on GGMs
sample.ggm <- bdgraph( data = data.sim, method = "ggm", iter = 10000 )

# Confusion Matrix for GGM method
conf.mat.plot( pred = sample.ggm, actual = data.sim )
```
covariance

## End(Not run)

covariance Estimated covariance matrix

Description

Provides the estimated covariance matrix.

Usage

covariance( bdgraph.obj, round = 2 )

Arguments

bdgraph.obj object of S3 class "bdgraph", from function bdgraph. It also can be an object of S3 class "ssgraph", from the function ssgraph::ssgraph() of R package ssgraph::ssgraph().

round value for rounding all probabilities to the specified number of decimal places.

Value

matrix which corresponds the estimated covariance matrix.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>

References


See Also

bdgraph, precision, plinks
Examples

```r
## Not run:
# Generating multivariate normal data from a 'circle' graph
data.sim <- bdgraph.sim( n = 70, p = 6, graph = "circle", vis = TRUE )

bdgraph.obj <- bdgraph( data = data.sim )
covariance( bdgraph.obj ) # Estimated covariance matrix
data.sim $ sigma # True covariance matrix
## End(Not run)
```

Discrete Weibull The Discrete Weibull Distribution (Type 1)

Description

Density, distribution function, quantile function and random generation for the discrete Weibull distribution (type I) with parameters \( q \) and \( \beta \).

Usage

```
ddweibull( x, q = exp( -1 ), beta = 1, zero = TRUE )
pdweibull( x, q = exp( -1 ), beta = 1, zero = TRUE )
qdweibull( p, q = exp( -1 ), beta = 1, zero = TRUE )
rdweibull( n, q = exp( -1 ), beta = 1, zero = TRUE )
```

Arguments

- **x**: vector of quantiles.
- **p**: vector of probabilities.
- **q, beta**: shape and scale parameters, the latter defaulting to 1.
- **zero**: logical; if TRUE (default), the support contains 0; FALSE otherwise.
- **n**: number of observations. If length(n) > 1, the length is taken to be the number required.

Details

The discrete Weibull distribution has density given by

\[
f(x) = q^x - q^{(x+1)}\beta, \quad x = 0, 1, 2, \ldots
\]

For the case \( \text{zero} = \text{FALSE} \):

\[
f(x) = q^{(x-1)}\beta - q^x, \quad x = 1, 2, \ldots
\]

Cumulative distribution function

\[
F(x) = 1 - q^{(x+1)}\beta
\]

For the case \( \text{zero} = \text{FALSE} \), \( x+1 \) should replaced by \( x \).
Value

dweibull gives the density, pdweibull gives the distribution function, qdweibull gives the quantile function, and rdweibull generates random values.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>, Pariya Behrouzi, Veronica Vinciotti

References


See Also
dweibull, bdw.reg, bdgraph.dw

Examples

n = 1000
q = 0.4
beta = 0.8

set.seed( 7 )
rdw = rdweibull( n = n, q = q, beta = beta )
plot( prop.table( table( rdw ) ), type = "h", col = "gray50" )
x = 0:max( rdw )
lines( x, dweibull( x = x, q = q, beta = beta ), type = "o", col = "blue", lwd = 2 )
hist( pdweibull( x = rdw, q = q, beta = beta ) )
plot( ecdf( rdw ) )
lines( x, dweibull( x, q = q, beta = beta ), col = "blue", lwd = 2, type = "s" )

---

**geneExpression**

**Human gene expression dataset**

Description

The dataset contains human gene expression of 100 transcripts (with unique Illumina TargetID) measured on 60 unrelated individuals.

Usage

data( geneExpression )
Format

The format is a matrix with 60 rows (number of individuals) and 100 column (number of transcripts).

References


Examples

data( geneExpression )
dim( geneExpression )
head( geneExpression )

| gnorm | Normalizing constant for G-Wishart |

Description

Calculates log of the normalizing constant of G-Wishart distribution based on the Monte Carlo method, developed by Atay-Kayis and Massam (2005).

Usage

gnorm( adj, b = 3, D = diag( ncol( adj ) ), iter = 100 )

Arguments

adj adjacency matrix corresponding to the graph structure. It is an upper triangular matrix in which \( a_{ij} = 1 \) if there is a link between notes \( i \) and \( j \), otherwise \( a_{ij} = 0 \).
b degree of freedom for G-Wishart distribution, \( W_G(b, D) \).
D positive definite \((p \times p)\) "scale" matrix for G-Wishart distribution, \( W_G(b, D) \). The default is an identity matrix.
iter number of iteration for the Monte Carlo approximation.

Details

Log of the normalizing constant approximation using Monte Carlo method for a G-Wishart distribution, \( K \sim W_G(b, D) \), with density:

\[
Pr(K) = \frac{1}{I(b, D)}|K|^{(b-2)/2} \exp \left\{ -\frac{1}{2} \text{trace}(K \times D) \right\}.
\]
graph.sim

Value

Log of the normalizing constant of G-Wishart distribution.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>

References

lihood in nondecomposable Gaussian graphical models, Biometrika, 92(2):317-335, doi:10.1093/
bimet/92.2.317

01621459.2021.1996377


See Also

rgwish, rwish

Examples

## Not run:
# adj: adjacency matrix of graph with 3 nodes and 2 links
adj <- matrix( c( 0, 0, 1,
                 0, 0, 1,
                 0, 0, 0 ), 3, 3, byrow = TRUE )

gnorm( adj, b = 3, D = diag( 3 ) )
## End(Not run)

describe

Description

Simulating undirected graph structures, including "random", "cluster", "scale-free", "lattice", "hub", "star", and "circle".

Usage

graph.sim( p = 10, graph = "random", prob = 0.2, size = NULL, class = NULL, vis = FALSE,
            rewire = 0.05 )
Arguments

- **p**: number of variables (nodes).
- **graph**: undirected graph with options "random", "cluster", "smallworld", "scale-free", "lattice", "hub", "star", and "circle". It also could be an adjacency matrix corresponding to a graph structure (an upper triangular matrix in which $g_{ij} = 1$ if there is a link between nodes $i$ and $j$, otherwise $g_{ij} = 0$).
- **prob**: if graph = "random", it is the probability that a pair of nodes has a link.
- **size**: number of links in the true graph (graph size).
- **class**: if graph = "cluster", it is the number of classes.
- **vis**: visualize the true graph structure.
- **rewire**: rewiring probability for smallworld network. Must be between 0 and 1.

Value

The adjacency matrix corresponding to the simulated graph structure, as an object with S3 class "graph".

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl> and Alexander Christensen

References


See Also

bdgraph.sim, bdgraph, bdgraph.mpl

Examples

```r
# Generating a 'hub' graph
adj <- graph.sim( p = 8, graph = "scale-free" )

plot( adj )

adj
```
**link2adj**

*Extract links from an adjacency matrix*

**Description**

Extract links from an adjacency matrix or an object of classes "sim" from function `bdgraph.sim` and "graph" from function `graph.sim`.

**Usage**

```r
link2adj( link, p = NULL )
```

**Arguments**

- `link` (2 × p) matrix or a data.frame corresponding to the links from the graph structure.
- `p` number of nodes of the graph.

**Value**

An adjacency matrix corresponding to a graph structure in which \( a_{ij} = 1 \) if there is a link between notes \( i \) and \( j \), otherwise \( a_{ij} = 0 \).

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**References**


**See Also**

`adj2link`, `graph.sim`

**Examples**

```r
# Generating a 'random' graph
adj <- graph.sim( p = 6, vis = TRUE )

link <- adj2link( adj )

link2adj( link, p = 6 )
```
### Description

Provides the estimated posterior probabilities for the most likely graphs or a specific graph.

### Usage

```r
pgraph( bdgraph.obj, number.g = 4, adj = NULL )
```

### Arguments

- `bdgraph.obj`: object of S3 class "bdgraph", from function `bdgraph`.
- `number.g`: number of graphs with the highest posterior probabilities to be shown. This option is ignored if `adj` is specified.
- `adj`: adjacency matrix corresponding to a graph structure. It is an upper triangular matrix in which \( a_{ij} = 1 \) if there is a link between notes \( i \) and \( j \), otherwise \( a_{ij} = 0 \). It also can be an object of S3 class "sim", from function `bdgraph.sim`.

### Value

- `selected_g`: adjacency matrices which corresponding to the graphs with the highest posterior probabilities.
- `prob_g`: vector of the posterior probabilities of the graphs corresponding to `selected_g`.

### Author(s)

Reza Mohammadi <a.mohammadi@uva.nl> and Ernst Wit

### References


### See Also

`bdgraph`, `bdgraph.mpl`
Examples

```r
## Not run:
# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 50, p = 6, size = 6, vis = TRUE )

bdgraph.obj <- bdgraph( data = data.sim, save = TRUE )

# Estimated posterior probability of the true graph
pgraph( bdgraph.obj, adj = data.sim )

# Estimated posterior probability of first and second graphs with highest probabilities
pgraph( bdgraph.obj, number.g = 2 )

## End(Not run)
```

---

**plinks**  
*Estimated posterior link probabilities*

Description

Provides the estimated posterior link probabilities for all possible links in the graph.

Usage

```r
plinks( bdgraph.obj, round = 2, burnin = NULL )
```

Arguments

- `bdgraph.obj`  
  object of S3 class "bdgraph", from function `bdgraph`. It also can be an object of S3 class "ssgraph", from the function `ssgraph::ssgraph()` of R package `ssgraph::ssgraph()`.
- `round`  
  value for rounding all probabilities to the specified number of decimal places.
- `burnin`  
  number of burn-in iteration to scape.

Value

An upper triangular matrix which corresponds the estimated posterior probabilities for all possible links.

Author(s)

Reza Mohammadi `<a.mohammadi@uva.nl>` and Ernst Wit
References


See Also

bdgraph, bdgraph.mpl

Examples

```r
## Not run:
# Generating multivariate normal data from a 'circle' graph
data.sim <- bdgraph.sim(n = 70, p = 6, graph = "circle", vis = TRUE )
bdgraph.obj <- bdgraph( data = data.sim, iter = 10000 )
plinks( bdgraph.obj, round = 2 )
## End(Not run)
```

plot.bdgraph

*Plot function for S3 class "bdgraph"*

Description

Visualizes structure of the selected graphs which could be a graph with links for which their estimated posterior probabilities are greater than 0.5 or graph with the highest posterior probability.

Usage

```r
## S3 method for class 'bdgraph'
plot( x, cut = 0.5, number.g = NULL, main = NULL, layout = igraph::layout_with_fr, vertex.size = 2, vertex.color = "orange", vertex.frame.color = "orange", vertex.label = NULL, vertex.label.dist = 0.5, vertex.label.color = "blue", edge.color = "lightblue", ... )
```
Arguments

- **x**
  object of S3 class "bdgraph", from function `bdgraph`.
- **cut**
  threshold for including the links in the selected graph based on the estimated posterior probabilities of the links; See the examples.
- **number.g**
  number of graphs with the highest probabilities. This option works for the case running function `bdgraph()` with option save = TRUE; See the examples.
- **main**
  Graphical parameter (see plot).
- **layout**
  vertex placement which is according to R package igraph; For different layouts, see layout of R package igraph.
- **vertex.size**
  vertex size which is according to R package igraph.
- **vertex.color**
  vertex color which is according to R package igraph.
- **vertex.frame.color**
  vertex frame color which is according to R package igraph.
- **vertex.label**
  vertex label. The default vertex labels are the vertex ids.
- **vertex.label.dist**
  vertex label distance which is according to R package igraph.
- **vertex.label.color**
  vertex label color which is according to R package igraph.
- **edge.color**
  edge color which is according to R package igraph.
- **...**
  additional plotting parameters. For the complete list, see igraph.plotting of R package igraph.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl> and Ernst Wit

References


See Also

`bdgraph`, `bdgraph.mpl`

Examples

```r
## Not run:
set.seed(100)

# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 100, p = 15, graph = "random", prob = 0.2, vis = TRUE )

bdgraph.obj <- bdgraph( data = data.sim )

plot( bdgraph.obj )
```

```
bdgraph.obj <- bdgraph( data = data.sim, save = TRUE )

plot( bdgraph.obj, cut = 0.5 )

plot( bdgraph.obj, number.g = 4 )

## End(Not run)

---

plot.graph

Plot function for S3 class "graph"

Description

Visualizes structure of the graph.

Usage

```
# S3 method for class 'graph'
plot( x, cut = 0.5, mode = "undirected", diag = FALSE, main = NULL,
     layout = igraph::layout_with_fr, vertex.size = 2, vertex.color = "orange",
     vertex.frame.color = "orange", vertex.label = NULL, vertex.label.dist = 0.5,
     vertex.label.color = "blue", edge.color = "lightblue", ... )
```

Arguments

- **x**: object of S3 class "graph", from function `graph.sim`.
- **cut**: for the case where input 'x' is the object of class "bdgraph" or "ssgraph". Threshold for including the links in the selected graph based on the estimated posterior probabilities of the links.
- **mode**: type of graph which is according to R package `igraph`.
- **diag**: logical which is according to R package `igraph`.
- **main**: graphical parameter (see plot).
- **layout**: vertex placement which is according to R package `igraph`; For different layouts, see `layout` of R package `igraph`.
- **vertex.size**: vertex size which is according to R package `igraph`.
- **vertex.color**: vertex color which is according to R package `igraph`.
- **vertex.frame.color**: vertex frame color which is according to R package `igraph`.
- **vertex.label**: vertex label. The default vertex labels are the vertex ids.
- **vertex.label.dist**: vertex label distance which is according to R package `igraph`.
- **vertex.label.color**: vertex label color which is according to R package `igraph`.
- **edge.color**: edge color which is according to R package `igraph`.
- **...**: additional plotting parameters. For the complete list, see `igraph.plotting` of R package `igraph`. 
plot.sim

Author(s)
Reza Mohammadi <a.mohammadi@uva.nl>

References

See Also
graph.sim, bdgraph.sim, plot.igraph

Examples

```r
# Generating a 'scale-free' graph
adj <- graph.sim( p = 20, graph = "scale-free" )
plot( adj )
```

Description
Plot function for S3 class "sim"

Usage
```
## S3 method for class 'sim'
plot( x, ... )
```

Arguments
- **x**: object of S3 class "sim", from function bdgraph.sim.
- **...**: additional plotting parameters. See plot.graph and for the complete list igraph.plotting of R package igraph.

Author(s)
Reza Mohammadi <a.mohammadi@uva.nl> and Ernst Wit

References

See Also
graph.sim, bdgraph.sim, plot.graph, plot.igraph
Examples

```r
set.seed(10)

# Generating synthetic multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim(n = 10, p = 15)

plot(data.sim)
```

---

**plotcoda**

**Convergence plot**

**Description**

Visualizes the cumulative occupancy fractions of all possible links in the graph. It can be used for monitoring the convergence of the sampling algorithms, BDMCMC and RJMCMC.

**Usage**

```r
plotcoda(bdgraph.obj, thin = NULL, control = TRUE, main = NULL,
         verbose = TRUE, ...)
```

**Arguments**

- `bdgraph.obj` object of S3 class "bdgraph", from function `bdgraph`. It also can be an object of S3 class "ssgraph", from the function `ssgraph::ssgraph()` of R package `ssgraph::ssgraph()`.
- `thin` option for getting fast result for a cumulative plot according to part of the iteration.
- `control` logical: if TRUE (default) and the number of nodes is greater than 15, then 100 links randomly is selected for visualization.
- `main` graphical parameter (see plot).
- `verbose` logical: if TRUE (default), report/print the calculation progress.
- `...` system reserved (no specific usage).

**Details**

Note that a spending time for this function depends on the number of nodes.
For fast result, you can choose bigger value for the 'thin' option.

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl>

**References**

plotroc

**See Also**

bdgraph, bdgraph.mpl, traceplot

**Examples**

```r
## Not run:
# Generating multivariate normal data from a 'circle' graph
data.sim <- bdgraph.sim( n = 50, p = 6, graph = "circle", vis = TRUE )

bdgraph.obj <- bdgraph( data = data.sim, iter = 10000, burnin = 0, save = TRUE )

plotcoda( bdgraph.obj )
## End(Not run)
```

---

**plotroc**  

**ROC plot**

**Description**

Draws the receiver operating characteristic (ROC) curve according to the true graph structure for object of S3 class "bdgraph", from function *bdgraph*.

**Usage**

```r
plotroc( pred, actual, cut = 200, smooth = FALSE, calibrate = TRUE,
        linetype = NULL, color = NULL, size = 1, main = "ROC Curve",
        xlab = "False Positive Rate", ylab = "True Positive Rate",
        legend = TRUE, legend.size = 17, legend.position = c( 0.7, 0.3 ),
        labels = NULL, auc = TRUE, theme = ggplot2::theme_minimal() )
```

**Arguments**

- **pred**: upper triangular matrix corresponding to the estimated posterior probabilities for all possible links. It can be an object with S3 class "bdgraph" from function *bdgraph*. It can be an object of S3 class "ssgraph", from the function ssgraph::ssgraph() of R package ssgraph::ssgraph(). It can be an object of S3 class "select", from the function huge.select of R package huge. Options est2, est3 and est4 are for comparing two or more different approaches.
- **actual**: adjacency matrix corresponding to the true graph structure in which $a_{ij} = 1$ if there is a link between notes $i$ and $j$, otherwise $a_{ij} = 0$. It can be an object with S3 class "sim" from function bdgraph.sim. It can be an object with S3 class "graph" from function graph.sim.
- **cut**: number of cut points.
- **smooth**: logical: for smoothing the ROC curve.
calibrate If TRUE, compute the value of AUC by taking the level of the probabilities into account.

linetype specification for the default plotting line type.
color specification for the default plotting color.
size specification for the default plotting line size.
main overall title for the plot.
xlab title for the x axis.
ylab title for the y axis.
legend logical: for adding legend to the ROC plot.
legend.size title for the x axis.
legend.position title for the y axis.
labels for legends of the legend to the ROC plot.
auc logical: to report AUC with legend.
theme theme for the plot from the function `ggplot2::ggplot()` of R package `ggplot2::ggplot()`.

Author(s)
Reza Mohammadi <a.mohammadi@uva.nl>; Lucas Vogels <l.f.o.vogels@uva.nl>

References

See Also
roc, pROC::plot.roc(), pROC::auc(), bdgraph, bdgraph.mpl, compare

Examples
```r
## Not run:
# To generate multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 30, p = 6, size = 7, vis = TRUE )

# To Run sampling algorithm
bdgraph.obj <- bdgraph( data = data.sim, iter = 10000 )

# To compare the results
plotroc( bdgraph.obj2j, data.sim )
```
# To compare the results based on CGGMs approach
bdgraph.obj2 <- bdgraph( data = data.sim, method = "gcgm", iter = 10000 )

# To Compare the results
plotroc( list( bdgraph.obj, bdgraph.obj2 ), data.sim, legend = FALSE )

## End(Not run)

posterior.predict  Posterior Predictive Samples

**Description**

Provides samples from the posterior predictive distribution.

**Usage**

```r
posterior.predict( object, iter = 1, ... )
```

**Arguments**

- `object`: object of S3 class "bdgraph", from function `bdgraph`.
- `iter`: number of predictions.
- `...`: additional parameters.

**Value**

A matrix containing the predicted datasets, corresponding to the samples from the joint posterior distribution.

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl> and Ernst Wit

**References**


See Also

bdgraph, bdgraph.mpl, bdgraph.dw

Examples

```r
## Not run:
# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 50, p = 6, size = 7, vis = TRUE )

bdgraph.obj <- bdgraph( data = data.sim )

posterior.predict( bdgraph.obj, iter = 20 )

## End(Not run)
```

### precision

**Estimated precision matrix**

Description

Provides the estimated precision matrix.

Usage

```
precision( bdgraph.obj, round = 2 )
```

Arguments

- `bdgraph.obj`: object of S3 class "bdgraph", from function `bdgraph`. It also can be an object of S3 class "ssgraph", from the function `ssgraph::ssgraph()` of R package `ssgraph::ssgraph()`.
- `round`: value for rounding all probabilities to the specified number of decimal places.

Value

matrix which corresponds the estimated precision matrix.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>

References


See Also

bdgraph, covariance, plinks
Examples

## Not run:
# Generating multivariate normal data from a 'circle' graph
data.sim <- bdgraph.sim( n = 70, p = 6, graph = "circle", vis = TRUE )

bdgraph.obj <- bdgraph( data = data.sim )

precision( bdgraph.obj ) # Estimated precision matrix

data.sim $ K # True precision matrix

## End(Not run)

predict.bdgraph Predict function for S3 class "bdgraph"

Description

Provides predict values of the results for function bdgraph.

Usage

## S3 method for class 'bdgraph'
predict( object, iter = 1, ... )

Arguments

object object of S3 class "bdgraph", from function bdgraph.
iter number of predictions.
... additional parameters.

Value

a matrix containing the predicted datasets, corresponding to the samples from the joint posterior distribution.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl> and Ernst Wit

References


See Also

`bdgraph`, `bdgraph.mpl`, `bdgraph.dw`

Examples

```r
## Not run:
# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 50, p = 6, size = 7, vis = TRUE )

bdgraph.obj <- bdgraph( data = data.sim )
predict( bdgraph.obj, iter = 20 )
## End(Not run)
```

print.bdgraph

*Print function for S3 class "bdgraph"*

Description

Prints the information about the selected graph which could be a graph with links for which their estimated posterior probabilities are greater than 0.5 or graph with the highest posterior probability. It provides adjacency matrix, size and posterior probability of the selected graph.

Usage

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

Arguments

- `x` object of S3 class "bdgraph", from function `bdgraph`.
- `...` system reserved (no specific usage).

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl> and Ernst Wit
References


See Also

bdgraph, bdgraph.mpl

Examples

## Not run:
# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 50, p = 6, vis = TRUE )

bdgraph.obj <- bdgraph( data = data.sim )

print( bdgraph.obj )
## End(Not run)

print.sim

Print function for S3 class "sim"

Description

Prints the information about the type of data, the sample size, the graph type, the number of nodes, number of links and sparsity of the true graph.

Usage

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

Arguments

x object of S3 class "sim", from function bdgraph.sim.

... system reserved (no specific usage).

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl> and Ernst Wit

References

See Also

graph.sim, bdgraph.sim

Examples

# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 20, p = 10, vis = TRUE )

print( data.sim )

Description

The dataset consist of 6 discrete variables as the potential risk factors of coronary heart disease. The data collected from 1841 men employed of a car factory in Czechoslovakia (Reinis et al. 1981).

Usage

data( reinis )

Format

The format is a matrix with 1841 rows (number of individuals) and 6 column (number of variables).

References


Examples

data( reinis )

summary( reinis )
rgwish

Sampling from G-Wishart distribution

Description

Generates random matrices, distributed according to the G-Wishart distribution with parameters $b$ and $D$, $W_G(b, D)$ with respect to the graph structure $G$. Note this function works for both non-decomposable and decomposable graphs.

Usage

rgwish( n = 1, adj = NULL, b = 3, D = NULL, threshold = 1e-8 )

Arguments

- **n**: number of samples required.
- **adj**: adjacency matrix corresponding to the graph structure which can be non-decomposable or decomposable. It should be an upper triangular matrix in which $a_{ij} = 1$ if there is a link between notes $i$ and $j$, otherwise $a_{ij} = 0$. adj could be an object of class "graph", from function graph.sim. It also could be an object of class "sim", from function bdgraph.sim. It also could be an object of class "bdgraph", from functions bdgraph.mpl or bdgraph.
- **b**: degree of freedom for G-Wishart distribution, $W_G(b, D)$.
- **D**: positive definite $(p \times p)$ "scale" matrix for G-Wishart distribution, $W_G(b, D)$. The default is an identity matrix.
- **threshold**: threshold value for the convergence of sampling algorithm from G-Wishart.

Details

Sampling from G-Wishart distribution, $K \sim W_G(b, D)$, with density:

$$Pr(K) \propto |K|^{(b-2)/2} \exp \left\{ -\frac{1}{2} \text{trace}(K \times D) \right\},$$

which $b > 2$ is the degree of freedom and $D$ is a symmetric positive definite matrix.

Value

A numeric array, say $A$, of dimension $(p \times p \times n)$, where each $A[i,j]$ is a positive definite matrix, a realization of the G-Wishart distribution, $W_G(b, D)$. Note, for the case $n = 1$, the output is a matrix.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl>
References


See Also
gnorm, rwish

Examples

```r
# Generating a 'circle' graph as a non-decomposable graph
adj <- graph.sim( p = 5, graph = "circle" )
adj # adjacency of graph with 5 nodes

sample <- rgwish( n = 1, adj = adj, b = 3, D = diag( 5 ) )
round( sample, 2 )

sample <- rgwish( n = 5, adj = adj )
round( sample, 2 )
```

---

**rmvnorm**

Generate data from the multivariate Normal distribution

Description

Random generation function from the multivariate Normal distribution with mean equal to `mean` and covariance matrix `sigma`.

Usage

```r
rmvnorm( n = 10, mean = rep( 0, length = ncol( sigma ) ),
        sigma = diag( length( mean ) ) )
```

Arguments

- `n` Number of observations.
- `mean` Mean vector, default is `rep(0, length = ncol(sigma))`.
- `sigma` positive definite covariance matrix, default is `diag(length(mean))`.

Value

A numeric matrix with rows equal to `n` and columns equal to `length(mean)`.
Build a ROC curve

This function builds a ROC curve specifically for graph structure learning and returns a “roc” object, a list of class “roc”. This object can be printed, plotted, or passed to the functions `pROC::roc()`, `pROC::ci()`, `pROC::smooth.roc()` and `pROC::coords()`. Additionally, two roc objects can be compared with `pROC::roc.test()`. This function is based on the roc function of R package pROC.

Usage

```r
roc( pred, actual, auc = TRUE, smooth = FALSE, plot = FALSE, quiet = TRUE, ... )
```

Arguments

- **pred**: adjacency matrix (or a vector) corresponding to an estimated graph. It can be an object with S3 class "bdgraph" from function `bdgraph`. It can be an object of S3 class "ssgraph", from the function `ssgraph::ssgraph()` of R package `ssgraph::ssgraph()`. It can be a numeric or ordered vector of the same length than actual, containing the predicted value of each observation.
- **actual**: adjacency matrix (or a vector) corresponding to the actual graph structure in which $a_{ij} = 1$ if there is a link between notes $i$ and $j$, otherwise $a_{ij} = 0$. It can be an object with S3 class "sim" from function `bdgraph.sim`. It can be an object with S3 class "graph" from function `graph.sim`. It can be a factor, numeric or character vector of responses (true class), typically encoded with 0 (controls) and 1 (cases). Only two classes can be used in a ROC curve.
- **smooth**: if TRUE, the ROC curve is passed to smooth to be smoothed.
- **auc**: compute the area under the curve (AUC)? If TRUE (default), additional arguments can be passed to `pROC::auc()`.
plot the ROC curve? If TRUE, additional arguments can be passed to \texttt{pROC::plot.roc}.

\texttt{quiet} if TRUE, turn off messages when direction and levels are auto-detected.

\texttt{...} further arguments to be passed to \texttt{pROC::roc}.

\textbf{Value}

If the data contained any NA value and \texttt{na.rm=FALSE}, NA is returned. Otherwise, if \texttt{smooth=FALSE}, a list of class “roc” with the following fields:

- \texttt{auc} if called with \texttt{auc=TRUE}, a numeric of class “auc” as defined in \texttt{pROC::auc}.
- \texttt{ci} if called with \texttt{ci=TRUE}, a numeric of class “ci” as defined in \texttt{pROC::ci}.
- \texttt{response} the response vector. Patients whose response is not \texttt{%in%} levels are discarded. If NA values were removed, a \texttt{na.action} attribute similar to \texttt{na.omit} stores the row numbers.
- \texttt{predictor} predictor vector converted to numeric as used to build the ROC curve. Patients whose response is not \texttt{%in%} levels are discarded. If NA values were removed, a \texttt{na.action} attribute similar to \texttt{na.omit} stores the row numbers.
- \texttt{original.predictor, original.response} response and predictor vectors as passed in argument.
- \texttt{levels} levels of the response as defined in argument.
- \texttt{controls} predictor values for the control observations.
- \texttt{cases} predictor values for the cases.
- \texttt{percent} if the sensitivities, specificities and AUC are reported in percent, as defined in argument.
- \texttt{direction} direction of the comparison, as defined in argument.
- \texttt{fun.sesp} function used to compute sensitivities and specificities. Will be re-used in bootstrap operations.
- \texttt{sensitivities} sensitivities defining the ROC curve.
- \texttt{specificities} specificities defining the ROC curve.
- \texttt{thresholds} thresholds at which the sensitivities and specificities were computed. See below for details.
- \texttt{call} how the function was called. See \texttt{match.call} for more details.

If \texttt{smooth=TRUE} a list of class “smooth.roc” as returned by \texttt{pROC::smooth}, with or without additional elements auc and ci (according to the call).

\textbf{Author(s)}

Reza Mohammadi <a.mohammadi@uva.nl>; Lucas Vogels <l.f.o.vogels@uva.nl>

\textbf{References}


See Also

plotroc, pROC::plot.roc(), pROC::auc(), pROC::print.roc(), bdgraph, bdgraph.mpl, compare

Examples

## Not run:
set.seed( 5 )

# Generating multivariate normal data from a 'scale-free' graph
data.sim = bdgraph.sim( n = 200, p = 15, graph = "scale-free", vis = TRUE )

# Running BDMCMC algorithm
sample.bdmcmc = bdgraph( data = data.sim, algorithm = "bdmcmc", iter = 10000 )

# ROC curve for BDMCMC algorithm
roc.bdmcmc = BDgraph::roc( pred = sample.bdmcmc, actual = data.sim, plot = TRUE )

# Running RJMCMC algorithm
sample.rjmcmc = bdgraph( data = data.sim, algorithm = "rjmcmc", iter = 10000 )

# ROC curve for RJMCMC algorithm
roc.rjmcmc = BDgraph::roc( pred = sample.rjmcmc, actual = data.sim, plot = TRUE )

# ROC curve for both BDMCMC and RJMCMC algorithms
pROC::ggroc( list( BDMCMC = roc.bdmcmc, RJMCMC = roc.rjmcmc ) )

## End(Not run)

---

rwish

Sampling from Wishart distribution

Description

Generates random matrices, distributed according to the Wishart distribution with parameters \( b \) and \( D, W(b, D) \).

Usage

rwish( n = 1, p = 2, b = 3, D = diag( p ) )

Arguments

n number of samples required.
p number of variables (nodes).
b degree of freedom for Wishart distribution, \( W(b, D) \).
D positive definite \((p \times p)\) "scale" matrix for Wishart distribution, \( W(b, D) \). The default is an identity matrix.
Details
Sampling from Wishart distribution, $K \sim W(b, D)$, with density:

$$Pr(K) \propto |K|^{(b-2)/2} \exp \left\{ -\frac{1}{2} \text{trace}(K \times D) \right\},$$

which $b > 2$ is the degree of freedom and $D$ is a symmetric positive definite matrix.

Value
A numeric array, say $A$, of dimension $(p \times p \times n)$, where each $A[i, i]$ is a positive definite matrix, a realization of the Wishart distribution $W(b, D)$. Note, for the case $n = 1$, the output is a matrix.

Author(s)
Reza Mohammadi <a.mohammadi@uva.nl>

References

See Also
gnorm, rgwish

Examples
```r
sample <- rwish( n = 3, p = 5, b = 3, D = diag( 5 ) )
round( sample, 2 )
```

---

select | Graph selection

Description
Provides the selected graph which, based on input, could be a graph with links for which their estimated posterior probabilities are greater than 0.5 (default) or a graph with the highest posterior probability; see examples.

Usage
```r
select( bdgraph.obj, cut = NULL, vis = FALSE )
```
Arguments

bdgraph.obj matrix in which each element response to the weight of the links. It can be an object of S3 class "bdgraph", from function bdgraph. It can be an object of S3 class "ssgraph", from the function ssgraph::ssgraph() of R package ssgraph::ssgraph().

cut threshold for including the links in the selected graph based on the estimated posterior probabilities of the links; see the examples.

vis visualize the selected graph structure.

Value

An adjacency matrix corresponding to the selected graph.

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl> and Ernst Wit

References


See Also

bdgraph, bdgraph.mpl

Examples

## Not run:
# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 50, p = 6, size = 7, vis = TRUE )

bdgraph.obj <- bdgraph( data = data.sim )

select( bdgraph.obj )

bdgraph.obj <- bdgraph( data = data.sim, save = TRUE )
sparsity

Compute the sparsity of a graph

Description
Compute the sparsity of a graph/network or an object of class "graph" from function `graph.sim` or an object of class "sim" from function `bdgraph.sim`.

Usage
```r
sparsity(adj)
```

Arguments
- `adj` adjacency matrix corresponding to a graph structure in which $a_{ij} = 1$ if there is a link between notes $i$ and $j$, otherwise $a_{ij} = 0$. It can be an object with S3 class "graph" from function `graph.sim`. It can be an object with S3 class "sim" from function `bdgraph.sim`.

Value
- value corresponding to the graph sparsity which is the proportion of the non-links (non-zero elements) in `adj`.

Author(s)
- Reza Mohammadi <a.mohammadi@uva.nl>

References

See Also
- `graph.sim`, `adj2link`, `link2adj`

Examples
```r
# Generating a 'random' graph
adj <- graph.sim( p = 10, graph = "random", prob = 0.4, vis = TRUE )

sparsity(adj)
```
**Description**

Provides a summary of the results for function `bdgraph`.

**Usage**

```r
## S3 method for class 'bdgraph'
summary(object, round = 2, vis = TRUE, ...)
```

**Arguments**

- `object`: object of S3 class "bdgraph", from function `bdgraph`.
- `round`: value for rounding all probabilities to the specified number of decimal places.
- `vis`: visualize the results.
- `...`: additional plotting parameters for the case `vis = TRUE`. See `plot.graph`.

**Value**

- `selected_g`: adjacency matrix corresponding to the selected graph which has the highest posterior probability.
- `p_links`: upper triangular matrix corresponding to the posterior probabilities of all possible links.
- `K_hat`: estimated precision matrix.

**Author(s)**

Reza Mohammadi <a.mohammadi@uva.nl> and Ernst Wit

**References**


**See Also**

`bdgraph`, `bdgraph.mpl`
The survey dataset concerns 1002 males in the U.S labor force, described by Hoff (2007). The seven observed variables which have been measured on various scales are as follow: the income (income), degree (degree), the number of children (children), parents income (pincome), parents degree (pdegree), number of parents children (pchildren), and age (age).

Usage

```r
data( surveyData )
```

Format

The format is a matrix with 1002 rows (number of individuals) and 7 column (number of variables).

References


Examples

```r
data( surveyData )
summary( surveyData )
```
Description

Trace plot for graph size for the objects of S3 class "bdgraph", from function \texttt{bdgraph}. It is a tool for monitoring the convergence of the sampling algorithms, BDMCMC and RJMCMC.

Usage

\begin{verbatim}
traceplot( bdgraph.obj, acf = FALSE, pacf = FALSE, main = NULL, ... )
\end{verbatim}

Arguments

- \texttt{bdgraph.obj} object of S3 class "bdgraph", from function \texttt{bdgraph}. It also can be an object of S3 class "ssgraph", from the function \texttt{ssgraph::ssgraph()} of R package \texttt{ssgraph::ssgraph()}.
- \texttt{acf} visualize the autocorrelation functions for graph size.
- \texttt{pacf} visualize the partial autocorrelations for graph size.
- \texttt{main} graphical parameter (see plot).
- \texttt{...} system reserved (no specific usage).

Author(s)

Reza Mohammadi <a.mohammadi@uva.nl> and Ernst Wit

References


See Also

\texttt{plotcoda, bdgraph, bdgraph.mpl}
## Not run:

```r
# Generating multivariate normal data from a 'random' graph
data.sim <- bdgraph.sim( n = 50, p = 6, size = 7, vis = TRUE )

bdgraph.obj <- bdgraph( data = data.sim, iter = 10000, burnin = 0, save = TRUE )

traceplot( bdgraph.obj )

traceplot( bdgraph.obj, acf = TRUE, pacf = TRUE )
```

## End(Not run)

### transfer

#### transfer for count data

#### Description

Transfers count data, by counting the duplicated rows.

#### Usage

```r
transfer( r_data )
```

#### Arguments

- `r_data`: \((n \times p)\) matrix or a `data.frame` corresponding to the data (n is the sample size and \(p\) is the number of variables).

#### Value

\((n \times p + 1)\) matrix of transferred data, in which the last column is the frequency of duplicated rows.

#### Author(s)

Reza Mohammadi (<a.mohammadi@uva.nl>) and Adrian Dobra

#### References


#### See Also

`bdgraph.mpl`, `bdgraph.sim`
Examples

# Generating multivariate binary data from a 'random' graph
data.sim <- bdgraph.sim( n = 12, p = 4, size = 4, type = "binary" )
r_data <- data.sim $ data
r_data

# Transfer the data
transfer( r_data )
Index

* Discrete Weibull
  - bdgraph.dw, 10
  - bdw.reg, 20
* Wishart
  - gnorm, 32
  - rgwish, 51
  - rwish, 55
* data preprocessing
  - bdgraph.npn, 16
  - transfer, 62
* datagen
  - bdgraph.sim, 17
* datasets
  - geneExpression, 31
  - reinis, 50
  - surveyData, 60
* distribution
  - Discrete Weibull, 30
  - gnorm, 32
  - rmvnorm, 52
  - rwish, 55
* graphs
  - adj2link, 4
  - graph.sim, 33
  - link2adj, 35
  - sparsity, 58
* hplot
  - plot.bdgraph, 38
  - plot.graph, 40
  - plot.sim, 41
  - plotcoda, 42
  - plotroc, 43
  - traceplot, 61
* iteration
  - bdgraph, 7
  - bdgraph.dw, 10
  - bdgraph.mpl, 13
  - bdw.reg, 20
  - gnorm, 32
* package
  - BDgraph-package, 2
* parameter learning
  - covariance, 29
  - precision, 46
* print
  - print.bdgraph, 48
  - print.sim, 49
* sampling algorithms
  - bdgraph, 7
  - bdgraph.dw, 10
  - bdgraph.mpl, 13
  - bdw.reg, 20
* simulation
  - bdgraph.sim, 17
  - graph.sim, 33
* structure learning
  - auc, 6
  - bdgraph, 7
  - bdgraph.dw, 10
  - bdgraph.mpl, 13
  - bf, 23
  - compare, 25
  - conf.mat, 26
  - conf.mat.plot, 27
  - pgraph, 36
  - plinks, 37
  - roc, 53
  - select, 56
  - %in%, 54
  - calc_joint_dist (BDgraph-package), 2
  - compute_measures (BDgraph-package), 2
  - compute_tp_fp (BDgraph-package), 2
  - ddweibull_reg (BDgraph-package), 2
  - detect_cores (BDgraph-package), 2
  - generate_clique_factors (BDgraph-package), 2
  - get_bounds_dw (BDgraph-package), 2
  - get_cores (BDgraph-package), 2
  - get_edges (BDgraph-package), 2
  - get_pairs (BDgraph-package), 2
INDEX

get_Ds_tgm_R (BDgraph-package), 2
get_g_prior (BDgraph-package), 2
get_g_start (BDgraph-package), 2
get_graph (BDgraph-package), 2
get_K_start (BDgraph-package), 2
get_S_n_p (BDgraph-package), 2
global_hc (BDgraph-package), 2
global_hc_binary (BDgraph-package), 2
hill_climb_mpl (BDgraph-package), 2
hill_climb_mpl_binary (BDgraph-package), 2
local_mb_hc (BDgraph-package), 2
local_mb_hc_binary (BDgraph-package), 2
log_mpl_binary (BDgraph-package), 2
log_mpl_discrete (BDgraph-package), 2
log_post_cond_dw (BDgraph-package), 2
near_positive_definite (BDgraph-package), 2
sample_ug (BDgraph-package), 2
update_mu_R (BDgraph-package), 2
update_tu_R (BDgraph-package), 2

adj2link, 4, 35, 58
auc, 6
BDgraph, 8, 11, 14
BDgraph (BDgraph-package), 2
BDgraph-package, 2
bdgraph.dw, 4, 9, 10, 15, 21, 31, 46, 48
bdgraph.mpl, 4, 7, 9, 13, 15, 16, 19, 24, 26, 34, 36, 38, 39, 43, 44, 46, 48, 49, 51, 55, 57, 59, 61
bdgraph.npn, 16
bdw.reg, 11, 13, 20, 31
beta, 20
bf, 23

compare, 4, 7, 9, 13, 15, 24, 25, 27, 28, 44, 55
conf.mat, 20, 28
conf.mat.plot, 27, 27
covariance, 29, 46
ddweibull, 21
ddweibull (Discrete Weibull), 30
Discrete Weibull, 30
dlnorm, 20
dnorm, 20
dunif, 20
dweibull, 31

geneExpression, 11, 20
ggplot2::ggplot(), 44
gnorm, 32, 52, 56
graph.sim, 4–6, 19, 24, 25, 27, 28, 33, 35, 40, 41, 43, 50, 51, 53, 58
huge, 25, 43
huge.select, 25, 43
igraph, 39, 40
igraph.plotting, 39–41
layout, 39, 40
link2adj, 5, 35, 58
match.call, 54
message, 54
model.frame, 11, 20
na.omit, 54
numeric, 6, 53
ordered, 6, 53
pdweibull (Discrete Weibull), 30
pgraph, 36
plinks, 29, 37, 46
plot.bdgraph, 38
plot.graph, 40, 41, 59
plot.igraph, 41
plot.sim, 41
plotcoda, 42, 61
plotroc, 7, 26, 43, 55
posterior.predict, 45
precision, 29, 46
predict.bdgraph, 47
print.bdgraph, 48
print.sim, 49
pROC::auc(), 7, 44, 53–55
pROC::ci(), 53, 54
pROC::coords(), 53
pROC::plot.roc(), 7, 44, 54, 55
pROC::print.roc(), 7, 55
pROC::roc(), 53, 54
pROC::roc.test(), 53
pROC::smooth(), 54
pROC::smooth.roc(), 53

qdweibull (Discrete Weibull), 30
rdweibull (Discrete Weibull), 30
reinis, 50
rgwish, 4, 33, 51, 53, 56
rmvnorm, 52
roc, 27, 28, 44, 53, 53
rwish, 33, 52, 53, 55

select, 56
smooth, 53
sparsity, 58
ssgraph::ssgraph(), 6, 8, 11, 14, 24–26, 28, 29, 37, 42, 43, 46, 53, 57, 61
summary.bdgraph, 9, 13, 15, 59
surveyData, 60

traceplot, 43, 61
transfer, 62