Package ‘huge’

April 1, 2020

Type Package
Title High-Dimensional Undirected Graph Estimation
Version 1.3.4.1
Author Haoming Jiang, Xinyu Fei, Han Liu, Kathryn Roeder, John Lafferty, Larry Wasserman, Xingguo Li, and Tuo Zhao
Maintainer Haoming Jiang <jianghm.ustc@gmail.com>
Depends R (>= 3.0.0)
Imports Matrix, igraph, MASS, grDevices, graphics, methods, stats, utils, Rcpp
LinkingTo Rcpp, RcppEigen
Description Provides a general framework for high-dimensional undirected graph estimation. It integrates data preprocessing, neighborhood screening, graph estimation, and model selection techniques into a pipeline. In preprocessing stage, the nonparanormal(npn) transformation is applied to help relax the normality assumption. In the graph estimation stage, the graph structure is estimated by Meinshausen-Buhlmann graph estimation or the graphical lasso, and both methods can be further accelerated by the lossy screening rule preselecting the neighborhood of each variable by correlation thresholding. We target on high-dimensional data analysis usually d >> n, and the computation is memory-optimized using the sparse matrix output. We also provide a computationally efficient approach, correlation thresholding graph estimation. Three regularization/thresholding parameter selection methods are included in this package: (1)stability approach for regularization selection (2) rotation information criterion (3) extended Bayesian information criterion which is only available for the graphical lasso.
License GPL-2
Repository CRAN
NeedsCompilation yes
R topics documented:

huge-package .................................................. 2
huge .......................................................... 4
huge.ct ......................................................... 7
huge.generator .................................................. 8
huge.glasso ..................................................... 10
huge.inference .................................................. 11
huge.mb ........................................................ 13
huge.npn ......................................................... 14
huge.plot ......................................................... 15
huge.roc ........................................................ 16
huge.select ..................................................... 17
huge.tiger ....................................................... 19
plot.huge ......................................................... 20
plot roc ........................................................ 21
plot.select ...................................................... 22
plot.sim ........................................................ 22
print.huge ....................................................... 23
print.roc ........................................................ 23
print.select .................................................... 24
print.sim ........................................................ 24
stockdata ......................................................... 25

Index 26

---

huge-package High-Dimensional Undirected Graph Estimation

Description

A package for high-dimensional undirected graph estimation

Details

Package: huge
Type: Package
Version: 1.2.7
Date: 2015-09-14
License: GPL-2
LazyLoad: yes
The package "huge" provides 8 main functions:
(1) the data generator creates random samples from multivariate normal distributions with different
graph structures. Please refer to huge.generator.
(2) the nonparanormal (npn) transformation helps relax the normality assumption. Please refer to
huge.npn.
(3) The correlation thresholding graph estimation. Please refer to huge.
(4) The Meinshausen-Buhlmann graph estimation. Please refer to huge.
(5) The graphical Lasso algorithm using lossless screening rule. Please refer and huge.

**Both (4) and (5) can be further accelerated by the lossy screening rule preselecting the neigh-
borhood of each node via thresholding sample correlation.
(6) The model selection using the stability approach to regularization selection. Please refer to
huge.select.
(7) The model selection using the rotation information criterion. Please refer to huge.select.
(8) The model selection using the extended Bayesian information criterion. Please refer to huge.select.

**Author(s)**

Tuo Zhao, Han Liu, Haoming Jiang, Kathryn Roeder, John Lafferty, and Larry Wasserman
Maintainers: Haoming Jiang<hjiang98@gatech.edu>;

**References**

1. T. Zhao and H. Liu. The huge Package for High-dimensional Undirected Graph Estimation in R.
2. H. Liu, F. Han, M. Yuan, J. Lafferty and L. Wasserman. High Dimensional Semiparametric
of Computational and Graphical Statistics, to appear, 2011. 4. Han Liu, Kathryn Roeder and
Larry Wasserman. Stability Approach to Regularization Selection (StARS) for High Dimensional
5. R. Foygel and M. Drton. Extended bayesian information criteria for gaussian graphical models.
7. J. Fan and J. Lv. Sure independence screening for ultra-high dimensional feature space (with
8. O. Banerjee, L. E. Ghaoui, A. d’Aspremont: Model Selection Through Sparse Maximum Likeli-
hood Estimation for Multivariate Gaussian or Binary Data. Journal of Machine Learning Research,
2008.
10. J. Friedman, T. Hastie and R. Tibshirani. Sparse inverse covariance estimation with the lasso,
11. N. Meinshausen and P. Buhlmann. High-dimensional Graphs and Variable Selection with the
see Also

huge.generator, huge.npn, huge.huge.plot and huge.roc

High-dimensional undirected graph estimation

Description

The main function for high-dimensional undirected graph estimation. Three graph estimation methods, including (1) Meinshausen-Buhlmann graph estimation (mb), (2) graphical lasso (glasso), (3) correlation thresholding graph estimation (ct) and (4) tuning-insensitive graph estimation (tiger), are available for data analysis.

Usage

huge(x, lambda = NULL, nlambda = NULL, lambda.min.ratio = NULL, method = "mb", scr = NULL, scr.num = NULL, cov.output = FALSE, sym = "or", verbose = TRUE)

Arguments

x
  There are 2 options: (1) x is an n by d data matrix (2) a d by d sample covariance matrix. The program automatically identifies the input matrix by checking the symmetry. (n is the sample size and d is the dimension).

lambda
  A sequence of decreasing positive numbers to control the regularization when method = "mb", "glasso" or "tiger", or the thresholding in method = "ct". Typical usage is to leave the input lambda = NULL and have the program compute its own lambda sequence based on nlambda and lambda.min.ratio. Users can also specify a sequence to override this. When method = "mb", "glasso" or "tiger", use with care - it is better to supply a decreasing sequence values than a single (small) value.

nlambda
  The number of regularization/thresholding parameters. The default value is 30 for method = "ct" and 10 for method = "mb", "glasso" or "tiger".

lambda.min.ratio
  If method = "mb", "glasso" or "tiger", it is the smallest value for lambda, as a fraction of the upperbound (MAX) of the regularization/thresholding parameter which makes all estimates equal to 0. The program can automatically generate lambda as a sequence of length = nlambda starting from MAX to lambda.min.ratio*MAX in log scale. If method = "ct", it is the largest sparsity level for estimated graphs. The program can automatically generate lambda as a sequence of length = nlambda, which makes the sparsity level of the graph path increases from 0 to lambda.min.ratio evenly. The default value is 0.1 when method = "mb", "glasso" or "tiger", and 0.05 method = "ct".

method
  Graph estimation methods with 4 options: "mb", "ct", "glasso" and "tiger". The default value is "mb".
If `scr = TRUE`, the lossy screening rule is applied to preselect the neighborhood before the graph estimation. The default value is FALSE. NOT applicable when `method = "ct", "mb", or "tiger"`.

The neighborhood size after the lossy screening rule (the number of remaining neighbors per node). ONLY applicable when `scr = TRUE`. The default value is `n-1`. An alternative value is `n/log(n)`. ONLY applicable when `scr = TRUE` and `method = "mb"`.

If `cov.output = TRUE`, the output will include a path of estimated covariance matrices. ONLY applicable when `method = "glasso"`. Since the estimated covariance matrices are generally not sparse, please use it with care, or it may take much memory under high-dimensional setting. The default value is FALSE.

Symmetrize the output graphs. If `sym = "and"`, the edge between node `i` and node `j` is selected ONLY when both node `i` and node `j` are selected as neighbors for each other. If `sym = "or"`, the edge is selected when either node `i` or node `j` is selected as the neighbor for each other. The default value is "or". ONLY applicable when `method = "mb"` or "tiger".

If `verbose = FALSE`, tracing information printing is disabled. The default value is TRUE.

The graph structure is estimated by Meinshausen-Buhlmann graph estimation or the graphical lasso, and both methods can be further accelerated via the lossy screening rule by preselecting the neighborhood of each variable by correlation thresholding. We target on high-dimensional data analysis usually `d » n`, and the computation is memory-optimized using the sparse matrix output. We also provide a highly computationally efficient approaches correlation thresholding graph estimation.

An object with S3 class "huge" is returned:

- `data`: The `n` by `d` data matrix or `d` by `d` sample covariance matrix from the input
- `cov.input`: An indicator of the sample covariance.
- `ind.mat`: The `scr.num` by `k` matrix with each column corresponding to a variable in `ind.group` and contains the indices of the remaining neighbors after the GSS. ONLY applicable when `scr = TRUE` and `approx = FALSE`.
- `lambda`: The sequence of regularization parameters used in `mb` or thresholding parameters in `ct`.
- `sym`: The `sym` from the input. ONLY applicable when `method = "mb"` or "tiger".
- `scr`: The `scr` from the input. ONLY applicable when `method = "mb"` or "glasso".
- `path`: A list of `k` by `k` adjacency matrices of estimated graphs as a graph path corresponding to `lambda`.
- `sparsity`: The sparsity levels of the graph path.
- `icov`: A list of `d` by `d` precision matrices as an alternative graph path (numerical path) corresponding to `lambda`. ONLY applicable when `method = "glasso"` or "tiger".
cov  A list of \( d \times d \) estimated covariance matrices corresponding to \( \lambda \). ONLY applicable when cov.output = TRUE and method = "glasso"

method  The method used in the graph estimation stage.

df  If method = "mb" or "tiger", it is a \( k \) by \( n_{\lambda} \) matrix. Each row contains the number of nonzero coefficients along the lasso solution path. If method = "glasso", it is a \( n_{\lambda} \) dimensional vector containing the number of nonzero coefficients along the graph path icov.

loglik  A \( n_{\lambda} \) dimensional vector containing the likelihood scores along the graph path (icov). ONLY applicable when method = "glasso". For an estimated inverse covariance \( Z \), the program only calculates \( \log(\det(Z)) - \text{trace}(SZ) \) where \( S \) is the empirical covariance matrix. For the likelihood for \( n \) observations, please multiply by \( n/2 \).

Note
This function ONLY estimates the graph path. For more information about the optimal graph selection, please refer to huge.select.

See Also
huge.generator, huge.select, huge.plot, huge.roc, and huge-package.

Examples

```r
#generate data
L = huge.generator(n = 50, d = 12, graph = "hub", g = 4)

#graph path estimation using mb
out1 = huge(L$data)
out1
plot(out1)  #Not aligned
plot(out1, align = TRUE)  #Aligned
huge.plot(out1$path[[3]])

#graph path estimation using the sample covariance matrix as the input.
#out1 = huge(cor(L$data), method = "glasso")
#out1
#plot(out1)  #Not aligned
#plot(out1, align = TRUE)  #Aligned
#huge.plot(out1$path[[3]])

#graph path estimation using ct
#out2 = huge(L$data, method = "ct")
#out2
#plot(out2)

#graph path estimation using glasso
#out3 = huge(L$data, method = "glasso")
#out3
```
huge.ct

#plot(out3)

#graph path estimation using tiger
#out4 = huge(L$data, method = "tiger")
#out4
#plot(out4)

---

huge.ct

Graph estimation via correlation thresholding (ct)

Description

See more details in huge

Usage

huge.ct(x, nlambda = NULL, lambda.min.ratio = NULL, lambda = NULL, verbose = TRUE)

Arguments

x
There are 2 options: (1) x is an n by d data matrix (2) a d by d sample covariance matrix. The program automatically identifies the input matrix by checking the symmetry. (n is the sample size and d is the dimension).

nlambda
The number of regularization/thresholding parameters. The default value is 30 for method = "ct" and 10 for method = "mb", "glasso" or "tiger".

lambda.min.ratio
If method = "mb", "glasso" or "tiger", it is the smallest value for lambda, as a fraction of the upperbound (MAX) of the regularization/thresholding parameter which makes all estimates equal to 0. The program can automatically generate lambda as a sequence of length = nlambda starting from MAX to lambda.min.ratio*MAX in log scale. If method = "ct", it is the largest sparsity level for estimated graphs. The program can automatically generate lambda as a sequence of length = nlambda, which makes the sparsity level of the graph path increases from 0 to lambda.min.ratio evenly. The default value is 0.1 when method = "mb", "glasso" or "tiger", and 0.05 method = "ct".

lambda
A sequence of decreasing positive numbers to control the regularization when method = "mb", "glasso" or "tiger", or the thresholding in method = "ct". Typical usage is to leave the input lambda = NULL and have the program compute its own lambda sequence based on nlambda and lambda.min.ratio. Users can also specify a sequence to override this. When method = "mb", "glasso" or "tiger", use with care - it is better to supply a decreasing sequence values than a single (small) value.

verbose
If verbose = FALSE, tracing information printing is disabled. The default value is TRUE.
See Also

huge, and huge-package.

---

### huge.generator

**Data generator**

**Description**

Implements the data generation from multivariate normal distributions with different graph structures, including "random", "hub", "cluster", "band" and "scale-free".

**Usage**

```r
huge.generator(n = 200, d = 50, graph = "random", v = NULL,
              u = NULL, g = NULL, prob = NULL, vis = FALSE, verbose = TRUE)
```

**Arguments**

- `n` The number of observations (sample size). The default value is 200.
- `d` The number of variables (dimension). The default value is 50.
- `graph` The graph structure with 4 options: "random", "hub", "cluster", "band" and "scale-free".
- `v` The off-diagonal elements of the precision matrix, controlling the magnitude of partial correlations with `u`. The default value is 0.3.
- `u` A positive number being added to the diagonal elements of the precision matrix, to control the magnitude of partial correlations. The default value is 0.1.
- `g` For "cluster" or "hub" graph, `g` is the number of hubs or clusters in the graph. The default value is about `d/20` if `d` >= 40 and 2 if `d` < 40. For "band" graph, `g` is the bandwidth and the default value is 1. NOT applicable to "random" graph.
- `prob` For "random" graph, it is the probability that a pair of nodes has an edge. The default value is `3/d`. For "cluster" graph, it is the probability that a pair of nodes has an edge in each cluster. The default value is `6*g/d` if `d/g` <= 30 and 0.3 if `d/g` > 30. NOT applicable to "hub" or "band" graphs.
- `vis` Visualize the adjacency matrix of the true graph structure, the graph pattern, the covariance matrix and the empirical covariance matrix. The default value is FALSE.
- `verbose` If `verbose = FALSE`, tracing information printing is disabled. The default value is TRUE.
Details

Given the adjacency matrix $\theta$, the graph patterns are generated as below:

(I) "random": Each pair of off-diagonal elements are randomly set $\theta[i,j]=\theta[j,i]=1$ for $i\neq j$ with probability $\text{prob}$, and $0$ otherwise. It results in about $d*(d-1)*\text{prob}/2$ edges in the graph.

(II) "hub": The row/columns are evenly partitioned into $g$ disjoint groups. Each group is associated with a "center" row $i$ in that group. Each pair of off-diagonal elements are set $\theta[i,j]=\theta[j,i]=1$ for $i\neq j$ if $j$ also belongs to the same group as $i$ and $0$ otherwise. It results in $d-g$ edges in the graph.

(III) "cluster": The row/columns are evenly partitioned into $g$ disjoint groups. Each pair of off-diagonal elements are set $\theta[i,j]=\theta[j,i]=1$ for $i\neq j$ with the probability $\text{prob}$ if both $i$ and $j$ belong to the same group, and $0$ otherwise. It results in about $g*(d/g)*(d/g-1)*\text{prob}/2$ edges in the graph.

(IV) "band": The off-diagonal elements are set to be $\theta[i,j]=1$ if $1<=|i-j|<=g$ and $0$ otherwise. It results in $(2d-1-g)*g/2$ edges in the graph.

(V) "scale-free": The graph is generated using B-A algorithm. The initial graph has two connected nodes and each new node is connected to only one node in the existing graph with the probability proportional to the degree of the each node in the existing graph. It results in $d$ edges in the graph.

The adjacency matrix $\theta$ has all diagonal elements equal to $0$. To obtain a positive definite precision matrix, the smallest eigenvalue of $\theta*v$ (denoted by $e$) is computed. Then we set the precision matrix equal to $\theta*v+(|e|+0.1+u)I$. The covariance matrix is then computed to generate multivariate normal data.

Value

An object with S3 class "sim" is returned:

- **data**: The $n$ by $d$ matrix for the generated data
- **sigma**: The covariance matrix for the generated data
- **omega**: The precision matrix for the generated data
- **sigmahat**: The empirical covariance matrix for the generated data
- **theta**: The adjacency matrix of true graph structure (in sparse matrix representation) for the generated data

See Also

huge and huge-package

Examples

```r
## band graph with bandwidth 3
L = huge.generator(graph = "band", g = 3)
```
plot(L)
## random sparse graph
L = huge.generator(vis = TRUE)

## random dense graph
L = huge.generator(prob = 0.5, vis = TRUE)

## hub graph with 6 hubs
L = huge.generator(graph = "hub", g = 6, vis = TRUE)

## hub graph with 8 clusters
L = huge.generator(graph = "cluster", g = 8, vis = TRUE)

## scale-free graphs
L = huge.generator(graph="scale-free", vis = TRUE)

### huge.glasso

The graphical lasso (glasso) using sparse matrix output

**Description**

See more details in huge

**Usage**

huge.glasso(x, lambda = NULL, lambda.min.ratio = NULL,
          nlambda = NULL, scr = NULL, cov.output = FALSE, verbose = TRUE)

**Arguments**

- **x**
  - There are 2 options: (1) x is an n by d data matrix (2) a d by d sample covariance matrix. The program automatically identifies the input matrix by checking the symmetry. (n is the sample size and d is the dimension).

- **lambda**
  - A sequence of decreasing positive numbers to control the regularization when method = "mb", "glasso" or "tiger", or the thresholding in method = "ct". Typical usage is to leave the input lambda = NULL and have the program compute its own lambda sequence based on nlambda and lambda.min.ratio. Users can also specify a sequence to override this. When method = "mb", "glasso" or "tiger", use with care - it is better to supply a decreasing sequence values than a single (small) value.

- **lambda.min.ratio**
  - If method = "mb", "glasso" or "tiger", it is the smallest value for lambda, as a fraction of the upperbound (MAX) of the regularization/thresholding parameter which makes all estimates equal to 0. The program can automatically generate lambda as a sequence of length = nlambda starting from MAX to lambda.min.ratio*MAX in log scale. If method = "ct", it is the largest sparsity level for estimated graphs. The program can automatically generate lambda as a
huge.inference

sequence of length = nlambdas, which makes the sparsity level of the graph path increases from 0 to lamda.min.ratio evenly. The default value is 0.1 when method = "mb", "glasso" or "tiger", and 0.05 method = "ct".

nlambdas

The number of regularization/thresholding parameters. The default value is 30 for method = "ct" and 10 for method = "mb", "glasso" or "tiger".

scr

If scr = TRUE, the lossy screening rule is applied to preselect the neighborhood before the graph estimation. The default value is FALSE. NOT applicable when method = "ct", "mb", or "tiger".

cov.output

If cov.output = TRUE, the output will include a path of estimated covariance matrices. ONLY applicable when method = "glasso". Since the estimated covariance matrices are generally not sparse, please use it with care, or it may take much memory under high-dimensional setting. The default value is FALSE.

verbose

If verbose = FALSE, tracing information printing is disabled. The default value is TRUE.

See Also

huge, and huge-package.

huge.inference

Graph inference

Description

Implements the inference for high dimensional graphical models, including Gaussian and Nonparanormal graphical models. We consider the problems of testing the presence of a single edge and the hypothesis is that the edge is absent.

Usage

huge.inference(data, T, adj, alpha = 0.05, type = "Gaussian", method = "score")

Arguments

data

The input n by d data matrix(n is the sample size and d is the dimension).

T

The estimated inverse of correlation matrix of the data.

adj

The adjacency matrix corresponding to the graph.

alpha

The significance level of hypothesis. The default value is 0.05.

type

The type of input data. There are 2 options: "Gaussian" and "Nonparanormal". The default value is "Gaussian".

method

When using nonparanormal graphical model. Test method with 2 options: "score" and "wald". The default value is "score".
Details

For Nonparanormal graphical model we provide Score test method and Wald Test. However it is really slow for inferencing on Nonparanormal model, especially for large data.

Value

An object is returned:

- **data**: The n by d data matrix from the input.
- **p**: The d by d p-value matrix of hypothesis.
- **error**: The type I error of hypothesis at alpha significance level.

References

1. Q Gu, Y Cao, Y Ning, H Liu. Local and global inference for high dimensional nonparanormal graphical models.

See Also

`huge` and `huge-package`.

Examples

```r
#generate data
L = huge.generator(n = 50, d = 12, graph = "hub", g = 4)

#graph path estimation using glasso
est = huge(L$data, method = "glasso")

#inference of Gaussian graphical model at 0.05 significance level
T = tail(est$icov, 1)[[1]]
out1 = huge.inference(L$data, T, L$theta)

#inference of Nonparanormal graphical model using score test at 0.05 significance level
T = tail(est$icov, 1)[[1]]
out2 = huge.inference(L$data, T, L$theta, type = "Nonparanormal")

#inference of Nonparanormal graphical model using wald test at 0.05 significance level
T = tail(est$icov, 1)[[1]]
out3 = huge.inference(L$data, T, L$theta, type = "Nonparanormal", method = "wald")

#inference of Nonparanormal graphical model using wald test at 0.1 significance level
T = tail(est$icov, 1)[[1]]
out4 = huge.inference(L$data, T, L$theta, 0.1, type = "Nonparanormal", method = "wald")
```
**huge.mb**

*Meinshausen & Buhlmann graph estimation*

**Description**

See more details in **huge**

**Usage**

```r
huge.mb(x, lambda = NULL, nlambdas = NULL, lambda.min.ratio = NULL,
scr = NULL, scr.num = NULL, idx.mat = NULL, sym = "or",
verbose = TRUE)
```

**Arguments**

- **x**
  - There are 2 options: (1) x is an n by d data matrix (2) a d by d sample covariance matrix. The program automatically identifies the input matrix by checking the symmetry. (n is the sample size and d is the dimension).

- **lambda**
  - A sequence of decreasing positive numbers to control the regularization when method = "mb", "glasso" or "tiger", or the thresholding in method = "ct". Typical usage is to leave the input lambda = NULL and have the program compute its own lambda sequence based on nlambda and lambda.min.ratio. Users can also specify a sequence to override this. When method = "mb", "glasso" or "tiger", use with care - it is better to supply a decreasing sequence values than a single (small) value.

- **nlambdas**
  - The number of regularization/thresholding parameters. The default value is 30 for method = "ct" and 10 for method = "mb", "glasso" or "tiger".

- **lambda.min.ratio**
  - If method = "mb", "glasso" or "tiger", it is the smallest value for lambda, as a fraction of the upperbound (MAX) of the regularization/thresholding parameter which makes all estimates equal to 0. The program can automatically generate lambda as a sequence of length = nlambdas starting from MAX to lambda.min.ratio*MAX in log scale. If method = "ct", it is the largest sparsity level for estimated graphs. The program can automatically generate lambda as a sequence of length = nlambdas, which makes the sparsity level of the graph path increases from 0 to lambda.min.ratio evenly. The default value is 0.1 when method = "mb", "glasso" or "tiger", and 0.05 method = "ct".

- **scr**
  - If scr = TRUE, the lossy screening rule is applied to preselect the neighborhood before the graph estimation. The default value is FALSE. NOT applicable when method = "ct", "mb", or "tiger".

- **scr.num**
  - The neighborhood size after the lossy screening rule (the number of remaining neighbors per node). ONLY applicable when scr = TRUE. The default value is n-1. An alternative value is n/log(n). ONLY applicable when scr = TRUE and method = "mb".

- **idx.mat**
  - Index matrix for screening.
Symmetrize the output graphs. If sym = "and", the edge between node i and node j is selected ONLY when both node i and node j are selected as neighbors for each other. If sym = "or", the edge is selected when either node i or node j is selected as the neighbor for each other. The default value is "or". ONLY applicable when method = "mb" or "tiger".

verbose If verbose = FALSE, tracing information printing is disabled. The default value is TRUE.

See Also

huge, and huge-package.

### huge.npn

**Nonparanormal(npn) transformation**

**Description**

Implements the Gaussianization to help relax the assumption of normality.

**Usage**

```r
huge.npn(x, npn.func = "shrinkage", npn.thresh = NULL, verbose = TRUE)
```

**Arguments**

- `x` The n by d data matrix representing n observations in d dimensions
- `npn.func` The transformation function used in the npn transformation. If npn.func = "truncation", the truncated ECDF is applied. If npn.func = "shrinkage", the shrunken ECDF is applied. The default is "shrinkage". If npn.func = "skeptic", the nonparanormal skeptic is applied.
- `npn.thresh` The truncation threshold used in nonparanormal transformation, ONLY applicable when npn.func = "truncation". The default value is \(1/(4\times(n^{0.25})\times\sqrt{\pi\times\log(n)})\).
- `verbose` If verbose = FALSE, tracing information printing is disabled. The default value is TRUE.

**Details**

The nonparanormal extends Gaussian graphical models to semiparametric Gaussian copula models. Motivated by sparse additive models, the nonparanormal method estimates the Gaussian copula by marginally transforming the variables using smooth functions. Computationally, the estimation of a nonparanormal transformation is very efficient and only requires one pass of the data matrix.
Value

data A d by d nonparanormal correlation matrix if npn.func = "skeptic", and a n by d data matrix representing n observations in d transformed dimensions otherwise.

See Also

huge and huge-package.

Examples

# generate nonparanormal data
L = huge.generator(graph = "cluster", g = 5)
L$data = L$data^5

# transform the data using the shrunken ECDF
Q = huge.npn(L$data)

# transform the non-Gaussian data using the truncated ECDF
Q = huge.npn(L$data, npn.func = "truncation")

# transform the non-Gaussian data using the truncated ECDF
Q = huge.npn(L$data, npn.func = "skeptic")

huge.plot

Graph visualization

Description

Implements the graph visualization using adjacency matrix. It can automatic organize 2D embedding layout.

Usage

huge.plot(G, epsflag = FALSE, graph.name = "default", cur.num = 1, location = NULL)

Arguments

G The adjacency matrix corresponding to the graph.
epsflag If epsflag = TRUE, save the plot as an eps file in the target directory. The default value is FALSE.
graph.name The name of the output eps files. The default value is "default".
cur.num The number of plots saved as eps files. Only applicable when epsflag = TRUE. The default value is 1.
location Target directory. The default value is the current working directory.
Details

The user can change cur.num to plot several figures and select the best one. The implementation is based on the popular package "igraph".

See Also

huge and huge-package.

Examples

```r
## visualize the hub graph
L = huge.generator(graph = "hub")
huge.plot(L$theta)

## visualize the band graph
L = huge.generator(graph = "band", g=5)
huge.plot(L$theta)

## visualize the cluster graph
L = huge.generator(graph = "cluster")
huge.plot(L$theta)

# show working directory
getwd()
# plot 5 graphs and save the plots as eps files in the working directory
huge.plot(L$theta, epsflag = TRUE, cur.num = 5)
```

draw.roc

Draw ROC Curve for a graph path

Description

Draws ROC curve for a graph path according to the true graph structure.

Usage

```r
huge.roc(path, theta, verbose = TRUE)
```

Arguments

- **path**: A graph path.
- **theta**: The true graph structure.
- **verbose**: If verbose = FALSE, tracing information printing is disabled. The default value is TRUE.

Details

To avoid the horizontal oscillation, false positive rates is automatically sorted in the ascent order and true positive rates also follow the same order.
Value

An object with S3 class "roc" is returned:

- **F1**: The F1 scores along the graph path.
- **tp**: The true positive rates along the graph path
- **fp**: The false positive rates along the graph paths
- **AUC**: Area under the ROC curve

Note

For a lasso regression, the number of nonzero coefficients is at most \( n-1 \). If \( d \gg n \), even when regularization parameter is very small, the estimated graph may still be sparse. In this case, the AUC may not be a good choice to evaluate the performance.

See Also

huge and huge-package.

Examples

```r
#generate data
L = huge.generator(d = 200, graph = "cluster", prob = 0.3)
out1 = huge(L$data)

#draw ROC curve
Z1 = huge.roc(out1$path,L$theta)

#Maximum F1 score
max(Z1$F1)
```

huge.select Model selection for high-dimensional undirected graph estimation

Description

Implements the regularization parameter selection for high dimensional undirected graph estimation. The optional approaches are rotation information criterion (ric), stability approach to regularization selection (stars) and extended Bayesian information criterion (ebic).

Usage

```r
huge.select(est, criterion = NULL, ebic.gamma = 0.5,
            stars.thresh = 0.1, stars.subsample.ratio = NULL, rep.num = 20,
            verbose = TRUE)
```
Arguments

est
An object with S3 class "huge".

dcriterion
Model selection criterion. "ric" and "stars" are available for all 3 graph estimation methods. ebic is only applicable when est$method = "glasso" in huge(). The default value is "ric".

ebic.gamma
The tuning parameter for ebic. The default value is 0.5. Only applicable when est$method = "glasso" and criterion = "ebic".

stars.thresh
The variability threshold in stars. The default value is 0.1. An alternative value is 0.05. Only applicable when criterion = "stars".

stars.subsample.ratio
The subsampling ratio. The default value is 10*sqrt(n)/n when n>144 and 0.8 when n<=144, where n is the sample size. Only applicable when criterion = "stars".

rep.num
The number of subsamplings when criterion = "stars" or rotations when criterion = "ric". The default value is 20. NOT applicable when criterion = "ebic".

verbose
If verbose = FALSE, tracing information printing is disabled. The default value is TRUE.

Details

Stability approach to regularization selection (stars) is a natural way to select optimal regularization parameter for all three estimation methods. It selects the optimal graph by variability of subsamplings and tends to overselect edges in Gaussian graphical models. Besides selecting the regularization parameters, stars can also provide an additional estimated graph by merging the corresponding subsampled graphs using the frequency counts. The subsampling procedure in stars may NOT be very efficient, we also provide the recent developed highly efficient, rotation information criterion approach (ric). Instead of tuning over a grid by cross-validation or subsampling, we directly estimate the optimal regularization parameter based on random Rotations. However, ric usually has very good empirical performances but suffers from underselctions sometimes. Therefore, we suggest if user are sensitive of false negative rates, they should either consider increasing r.num or applying the stars to model selection. Extended Bayesian information criterion (ebic) is another competitive approach, but the ebic.gamma can only be tuned by experience.

Value

An object with S3 class "select" is returned:

refit
The optimal graph selected from the graph path

opt.icov
The optimal precision matrix from the path only applicable when method = "glasso"

opt.cov
The optimal covariance matrix from the path only applicable when method = "glasso" and est$cov is available.

merge
The graph path estimated by merging the subsampling paths. Only applicable when the input criterion = "stars".
The model selection is NOT available when the data input is the sample covariance matrix.

**See Also**

*huge* and *huge-package*.

**Examples**

```r
#generate data
L = huge.generator(d = 20, graph="hub")
out.mb = huge(L$data)
out.ct = huge(L$data, method = "ct")
out.glasso = huge(L$data, method = "glasso")

#model selection using ric
out.select = huge.select(out.mb)
plot(out.select)

#model selection using stars
#out.select = huge.select(out.ct, criterion = "stars", stars.thresh = 0.05, rep.num=10)
#plot(out.select)

#model selection using ebic
out.select = huge.select(out.glasso,criterion = "ebic")
plot(out.select)
```

---

**huge.tiger**  
*Tuning-insensitive graph estimation*

**Description**

See more details in *huge*.
Usage

`huge.tiger(x, lambda = NULL, nlambda = NULL, lambda.min.ratio = NULL, sym = "or", verbose = TRUE)`

Arguments

- **x**: There are 2 options: (1) `x` is an `n` by `d` data matrix (2) a `d` by `d` sample covariance matrix. The program automatically identifies the input matrix by checking the symmetry. (`n` is the sample size and `d` is the dimension).

- **lambda**: A sequence of decreasing positive numbers to control the regularization when `method = "mb", "glasso" or "tiger"`, or the thresholding in `method = "ct"`. Typical usage is to leave the input `lambda = NULL` and have the program compute its own lambda sequence based on `nlambda` and `lambda.min.ratio`. Users can also specify a sequence to override this. When `method = "mb", "glasso" or "tiger"`, use with care - it is better to supply a decreasing sequence values than a single (small) value.

- **nlambda**: The number of regularization/thresholding parameters. The default value is 30 for `method = "ct"` and 10 for `method = "mb", "glasso" or "tiger"`.

- **lambda.min.ratio**: If `method = "mb", "glasso" or "tiger"`, it is the smallest value for `lambda`, as a fraction of the upperbound (MAX) of the regularization/thresholding parameter which makes all estimates equal to 0. The program can automatically generate lambda as a sequence of length = `nlambda` starting from MAX to `lambda.min.ratio*MAX` in log scale. If `method = "ct"`, it is the largest sparsity level for estimated graphs. The program can automatically generate lambda as a sequence of length = `nlambda`, which makes the sparsity level of the graph path increases from 0 to `lambda.min.ratio` evenly. The default value is 0.1 when `method = "mb", "glasso" or "tiger"`, and 0.05 method = "ct".

- **sym**: Symmetrize the output graphs. If `sym = "and"`, the edge between node `i` and node `j` is selected ONLY when both node `i` and node `j` are selected as neighbors for each other. If `sym = "or"`, the edge is selected when either node `i` or node `j` is selected as the neighbor for each other. The default value is "or". ONLY applicable when `method = "mb" or "tiger"`.

- **verbose**: If `verbose = FALSE`, tracing information printing is disabled. The default value is `TRUE`.

See Also

`huge`, and `huge-package`.

**plot.huge**

*Plot function for S3 class "huge"*

Description

Plot sparsity level information and 3 typical sparse graphs from the graph path.
Usage

## S3 method for class 'huge'
plot(x, align = FALSE, ...)

Arguments

x         An object with S3 class "huge"
align     If align = FALSE, 3 plotted graphs are aligned
...
System reserved (No specific usage)

See Also

huge

---

plot.roc

Plot function for S3 class "roc"

---

Description

Plot the ROC curve for an object with S3 class "roc".

Usage

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

Arguments

x         An object with S3 class "roc"
...
System reserved (No specific usage)

See Also

huge.roc
plot.select  

Plot function for S3 class "select"

Description
Plot the optimal graph by model selection.

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

Arguments
x An object with S3 class "select"
...

See Also
huge.select

plot.sim  

Plot function for S3 class "sim"

Description
Visualize the covariance matrix, the empirical covariance matrix, the adjacency matrix and the graph pattern of the true graph structure.

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

Arguments
x An object with S3 class "sim"
...

See Also
huge.generator and huge
print.huge

Print function for S3 class "huge"

Description
Print the information about the model usage, the graph path length, graph dimension, sparsity level.

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

Arguments
x An object with S3 class "huge".
...
System reserved (No specific usage)

See Also
huge

print.roc

Print function for S3 class "roc"

Description
Print the information about true positive rates, false positive rates, the area under curve and maximum F1 score.

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

Arguments
x An object with S3 class "roc".
...
System reserved (No specific usage)

See Also
huge.roc
**print.select**  
*Print function for S3 class "select"*

**Description**
Print the information about the model usage, graph dimension, model selection criterion, sparsity level of the optimal graph.

**Usage**
```
## S3 method for class 'select'
print(x, ...)
```

**Arguments**
- `x` An object with S3 class "select".
- `...` System reserved (No specific usage)

**See Also**
- huge.select

---

**print.sim**  
*Print function for S3 class "sim"*

**Description**
Print the information about the sample size, the dimension, the pattern and sparsity of the true graph structure.

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

**Arguments**
- `x` An object with S3 class "sim".
- `...` System reserved (No specific usage)

**See Also**
- huge.generator
Description
This data set consists of stock price and company information.

Usage
data(stockdata)

Format
The format is a list containing contains two matrices. 1. data - 1258x452, represents the 452 stocks’ close prices for 1258 trading days. 2. info - 452x3: The 1st column: the query symbol for each company. The 2nd column: the category for each company. The 3rd column: the full name of each company.

Details
This data set can be used to perform high-dimensional graph estimation to analyze the relationships between S&P 500 companies.

Source
It was publicly available at finance.yahoo, which is now out of date

Examples
data(stockdata)
image(stockdata$data)
stockdata$info
Index

*Topic datasets
  stockdata, 25
_PACKAGE (huge-package), 2

huge, 3, 4, 4, 7–17, 19–23
huge-package, 2
huge.ct, 7
huge.generator, 3, 4, 6, 8, 22, 24
huge.glasso, 10
huge.inference, 11
huge.mb, 13
huge.npn, 3, 4, 14
huge.plot, 4, 6, 15
huge.roc, 4, 6, 16, 21, 23
huge.select, 3, 6, 17, 22, 24
huge.tiger, 19

plot.huge, 20
plot.roc, 21
plot.select, 22
plot.sim, 22
print.huge, 23
print.roc, 23
print.select, 24
print.sim, 24

stockdata, 25