Package: huge (via r-universe)

October 20, 2026

Type Package

Title High-Dimensional Undirected Graph Estimation

Version 2.0.1

Maintainer Tuo Zhao <tourzhao@gatech.edu>

Depends R (>= 3.4.0)

Imports Matrix, igraph, MASS, grDevices, graphics, methods, parallel, stats, utils, Rcpp

Suggests testthat (>= 3.1.7)

LinkingTo Rcpp

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, the graphical lasso, or the TIGER (tuning-insensitive graph estimation and regression) method, and the first two 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

URL https://github.com/Gatech-Flash/huge

BugReports https://github.com/Gatech-Flash/huge/issues

NeedsCompilation yes

SystemRequirements C++17

RoxygenNote 7.3.3

Encoding UTF-8

Author Haoming Jiang [aut], Xinyu Fei [aut], Han Liu [aut], Kathryn Roeder [aut], John Lafferty [aut], Larry Wasserman [aut], Xingguo Li [aut], Tuo Zhao [aut, cre]

Repository https://cran.r-universe.dev

Date/Publication 2026-08-04 10:30:34 UTC

RemoteUrl https://github.com/cran/huge

RemoteRef HEAD

RemoteSha ca32459e0e05fc1691d0088bd1df5a561e7aef9e

Contents

huge-package . . . 3
huge . . . 4
huge.ct . . . 7
huge.generator . . . 8
huge.glasso . . . 11
huge.inference . . . 12
huge.mb . . . 14
huge.npn . . . 15
huge.plot . . . 16
huge.roc . . . 18
huge.select . . . 19
huge.tiger . . . 21
plot.huge . . . 23
plot.roc . . . 23
plot.select . . . 24
plot.sim . . . 24
print.huge . . . 25
print.roc . . . 25
print.select . . . 26
print.sim . . . 26
stockdata . . . 27

Index **28**

huge-package *High-Dimensional Undirected Graph Estimation*

Description

A package for high-dimensional undirected graph estimation

Details

The package "huge" provides 9 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 to `huge`.
(6) The tuning-insensitive graph estimation (tiger). Please refer to `huge.tiger`.

Both (4) and (5) can be further accelerated by the lossy screening rule preselecting the neighborhood of each node via thresholding sample correlation.
(7) The model selection using the stability approach to regularization selection. Please refer to `huge.select`.
(8) The model selection using the rotation information criterion. Please refer to `huge.select`.
(9) 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
Maintainer: Tuo Zhao <tourzhao@gatech.edu>

References

1. T. Zhao and H. Liu. The huge Package for High-dimensional Undirected Graph Estimation in R. *Journal of Machine Learning Research*, 2012
2. H. Liu, F. Han, M. Yuan, J. Lafferty and L. Wasserman. High Dimensional Semiparametric Gaussian Copula Graphical Models. *Annals of Statistics*,2012
3. D. Witten and J. Friedman. New insights and faster computations for the graphical lasso. *Journal 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 Graphical Models. *Advances in Neural Information Processing Systems*, 2010.
5. R. Foygel and M. Drton. Extended bayesian information criteria for gaussian graphical models. *Advances in Neural Information Processing Systems*, 2010.
6. H. Liu, J. Lafferty and L. Wasserman. The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs. *Journal of Machine Learning Research*, 2009
7. J. Fan and J. Lv. Sure independence screening for ultra-high dimensional feature space (with

discussion). *Journal of Royal Statistical Society B*, 2008.
8. O. Banerjee, L. E. Ghaoui, A. d'Aspremont: Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data. *Journal of Machine Learning Research*, 2008.
9. J. Friedman, T. Hastie and R. Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 2008.
10. J. Friedman, T. Hastie and R. Tibshirani. Sparse inverse covariance estimation with the lasso, *Biostatistics*, 2007.
11. N. Meinshausen and P. Buhlmann. High-dimensional Graphs and Variable Selection with the Lasso. *The Annals of Statistics*, 2006.

See Also

`huge.generator`, `huge.npn`, `huge`, `huge.tiger`, `huge.select`, `huge.plot` and `huge.roc`

---

huge *High-dimensional undirected graph estimation*

Description

The main function for high-dimensional undirected graph estimation. Four 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,
  input.type = "auto"
)
```

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 — An optional numeric scalar or non-empty one-dimensional numeric path. For method = "mb", "glasso", or "tiger", values must be finite, strictly positive, and non-increasing; ties are allowed. For method = "ct", values must be finite and non-negative, zero is allowed, and thresholds are applied in the supplied order. Leave lambda = NULL to generate a path from nlambda and lambda.min.ratio.

nlambda — The number of regularization/thresholding parameters. The default value is 20 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 when method = "ct".

method — Graph estimation methods with 4 options: "mb", "ct", "glasso" and "tiger". The default value is "mb".

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" 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".

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.

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.

input.type — How to interpret x: "auto" preserves symmetry-based detection, "data" forces an observation matrix, and "covariance" requires a square covariance or correlation matrix. Use "data" for square symmetric observation matrices. For glasso covariance input, "auto" retains the historical diagonal-sensitive default lambda scale; use "covariance" for a scale based only on off-diagonal entries. Glasso may accept an indefinite pairwise covariance estimate when regularization produces a certified positive-definite result; CT, MB, and TIGER require positive semidefiniteness.

Details

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.

Value

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 d matrix with each column containing the indices of the remaining neighbors of the corresponding variable after the lossy screening. ONLY applicable when scr = TRUE.
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 d by d 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 by 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 d by nlambda matrix. Each row contains the number of nonzero coefficients along the lasso solution path. If method = "glasso", it is a nlambda dimensional vector containing the number of nonzero coefficients along the graph path icov.
loglik	A nlambda 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)) - 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

```
#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]])
```

---

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,
  input.type = "auto"
)
```

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 20 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 when method = "ct".

lambda — A numeric scalar or non-empty one-dimensional numeric input of finite, non-negative thresholds. Values are applied in the supplied order, and zero is allowed. Leave lambda = NULL to generate a path from nlambda and lambda.min.ratio.

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

input.type — How to interpret x: "auto" preserves symmetry-based detection, "data" forces an observation matrix, and "covariance" requires a square covariance or correlation matrix.

Details

The default path targets increasing numbers of undirected edges and defines each graph by the strict rule abs(correlation) > lambda. Equal-weight edges are never split, so ties can make the realized sparsity smaller than the nominal target. Reusing the returned lambda values therefore reconstructs the same path. When lambda = NULL, nlambda must be a positive integer and lambda.min.ratio must lie in (0, 1]. Supplying lambda overrides those two arguments.

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

```
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), as an integer of at least 2. The default value is 200.
d	The positive integer number of variables (dimension). The default value is 50.
graph	The graph structure with 5 options: "random", "hub", "cluster", "band" and "scale-free".
v	A finite positive number used for the off-diagonal elements of the precision matrix, controlling the magnitude of partial correlations with u. The default value is 0.3.
u	A finite positive number added to the diagonal elements of the precision matrix to control the magnitude of partial correlations. The default value is 0.1.
g	A finite positive integer. For "cluster" or "hub" graph, g is the number of clusters or hubs; values above d are treated as d. The default is about d/20 if d >= 40 and min(2, d) otherwise. For "band" graph, g is the bandwidth and defaults to 1; values above d - 1 add no edges. Not applicable to "random" or "scale-free" graph.
prob	For "random" or "cluster" graph, a finite number in [0, 1] giving the probability that a pair of nodes has an edge. The default is min(1, 3/d) for "random". For "cluster", the default is min(1, 6*g/d) if d/g <= 30 and 0.3 otherwise. Not applicable to "hub", "band", or "scale-free" graph.
vis	A single non-missing logical. If TRUE, visualize the adjacency matrix, graph pattern, covariance matrix, and empirical correlation matrix. The default is FALSE.
verbose	A single non-missing logical. If FALSE, tracing output is disabled. The default 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!=j with probability prob, and 0 otherwise. It results in about d*(d-1)*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!=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!=j with the probability probif both i and j belong to the same group, and 0 otherwise. It results in about g*(d/g)*(d/g-1)*prob/2 edges in the graph.

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

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. Every graph type returns an empty graph with zero sparsity when d=1.

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 correlation matrix for the generated data
theta	The adjacency matrix of true graph structure (in sparse matrix representation) for the generated data
sparsity	The proportion of possible off-diagonal entries present in the graph; zero when d=1

See Also

huge and huge-package

Examples

```
## 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,
  input.type = "auto"
)
```

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 numeric scalar or non-empty one-dimensional numeric input defining a finite, strictly positive, non-increasing regularization path; tied values are allowed. Leave lambda = NULL to generate a path from nlambda and lambda.min.ratio.
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 when method = "ct".
nlambda	The number of regularization/thresholding parameters. The default value is 20 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.
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.
input.type	How to interpret x: "auto" preserves symmetry-based detection, "data" forces an observation matrix, and "covariance" requires a square covariance or correlation matrix. For covariance input with lambda = NULL, "auto" retains the historical diagonal-sensitive lambda scale; use "covariance" for a scale based only on off-diagonal entries. Indefinite pairwise covariance estimates are accepted only when regularization produces a certified positive-definite result.

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	A finite numeric n by d data matrix with at least two observations and no constant columns.
T	A finite d by d estimate of the inverse correlation matrix with a positive diagonal.
adj	A finite numeric or logical d by d adjacency matrix corresponding to the graph.
alpha	The significance level in (0, 1]. 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	For a Nonparanormal model, the test method: "score" or "wald". The default is "score". Ignored for Gaussian inference.

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. Gaussian inference supports one variable, while Nonparanormal inference requires at least two. Nonparanormal score-test diagonal p-values do not represent edges and may be undefined; every tested off-diagonal p-value must be finite.

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.
2.J Jankova, S Van De Geer. Confidence intervals for high-dimensional inverse covariance estimation. *Electronic Journal of Statistics*, 2015.

See Also

huge, and huge-package.

Examples

```
#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

```
huge.mb(
  x,
  lambda = NULL,
  nlambda = NULL,
  lambda.min.ratio = NULL,
  scr = NULL,
  scr.num = NULL,
  idx.mat = NULL,
  sym = "or",
  verbose = TRUE,
  input.type = "auto"
)
```

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 numeric scalar or non-empty one-dimensional numeric input defining a finite, strictly positive, non-increasing regularization path; tied values are allowed. Leave lambda = NULL to generate a path from nlambda and lambda.min.ratio.
nlambda	The number of regularization/thresholding parameters. The default value is 20 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 when 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.

scr.num	The neighborhood size after the lossy screening rule (the number of remaining neighbors per node). It must be an integer between 1 and d - 1. 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	A scr.num by d index matrix for screening. Column m must contain distinct, zero-based indices in 0:(d - 1) and must exclude the response index m - 1.
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.
input.type	How to interpret x: "auto" preserves symmetry-based detection, "data" forces an observation matrix, and "covariance" requires a square covariance or correlation matrix.

See Also

huge, and huge-package.

---

huge.npn *Nonparanormal(npn) transformation*

Description

Implements the Gaussianization to help relax the assumption of normality.

Usage

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

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*(n^0.25)*sqrt(pi*log(n))).
verbose	If verbose = FALSE, tracing information printing is disabled. The default value is TRUE.
na.last	for controlling the treatment of NAs. If TRUE, missing values in the data are put last; if FALSE, they are put first; if NA, they are removed; if "keep" they are kept with rank NA. See also rank.

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")

# estimate the nonparanormal correlation matrix using the skeptic estimator
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 applicale 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

```
## 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)

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

huge.roc *Draw ROC Curve for a graph path*

Description

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

Usage

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

Arguments

path	A graph path.
theta	The true graph structure, containing at least one edge and one absent off-diagonal edge.
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

ROC/AUC is undefined when theta contains only edges or only non-edges, so those one-class truth matrices are rejected. For a lasso regression, the number of nonzero coefficients is at most n-1. If d>>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

```
#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

```
huge.select(
  est,
  criterion = NULL,
  ebic.gamma = 0.5,
  stars.thresh = 0.1,
  stars.subsample.ratio = NULL,
  rep.num = 20,
  verbose = TRUE,
  num.cores = 1
)
```

Arguments

`est`	An object with S3 class `"huge"`.
`criterion`	Model selection criterion. `"ric"` is available for all four estimation methods, `"stars"` for `"mb"`, `"ct"`, and `"glasso"`, and `"ebic"` only for `"glasso"`. Defaults are `"ric"` for `"mb"` and `"tiger"`, `"stars"` for `"ct"`, and `"ebic"` for `"glasso"`.
`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`.

`num.cores`
: The number of CPU cores used to fit the stars subsamplings in parallel via `parallel::mclapply`. The default value is `1` (serial). At most two forked workers are used, and this argument is ignored on Windows. Each forked worker limits huge's native OpenMP code to one thread, but an external threaded BLAS may still create additional threads; `num.cores = 1` is the portable choice when a bounded thread budget or fork safety matters. Results are identical to the serial path for the same random seed. Only applicable when `criterion = "stars"`.

Details

Stability approach to regularization selection (stars) selects the optimal graph by variability of subsamplings and tends to overselect edges in Gaussian graphical models. It is available for `"mb"`, `"ct"`, and `"glasso"`. TIGER can certify different lambda-path prefixes on different subsamples, so TIGER with stars is rejected until a common certified-prefix protocol is available; use `"ric"` for TIGER. 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 underselections sometimes. Therefore, we suggest if user are sensitive of false negative rates, they should either consider increasing `rep.num` or applying the stars to model selection where supported. Extended Bayesian information criterion (ebic) is another competitive approach, but the `ebic.gamma` can only be tuned by experience.

For `criterion = "stars"`, `est$lambda` must be non-increasing; tied values are allowed.

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"`.

`variability`
: The variability along the subsampling paths. Only applicable when the input `criterion = "stars"`.

ebic.score	Extended BIC scores for regularization parameter selection. Only applicable when criterion = "ebic".
opt.index	The index of the selected regularization parameter. NOT applicable when the input criterion = "ric"
opt.lambda	The selected regularization/thresholding parameter.
opt.sparsity	The sparsity level of "refit".

and anything else included in the input est

Note

The model selection is NOT available when the data input is the sample covariance matrix.

See Also

huge and huge-package.

Examples

```
#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,
  input.type = "auto"
)
```

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 numeric scalar or non-empty one-dimensional numeric input defining a finite, strictly positive, non-increasing regularization path. Tied values are allowed. Leave lambda = NULL to generate the path from nlambda and lambda.min.ratio in C++.
nlambda	The number of regularization/thresholding parameters. The default value is 20 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 when 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.
input.type	How to interpret x: "auto" preserves symmetry-based detection, "data" forces an observation matrix, and "covariance" requires a square covariance or correlation matrix. Correlation construction, covariance validation, and automatic lambda selection then occur together in C++.

Details

Raw observations are centered and normalized, while covariance input is converted to a correlation matrix, inside the shared C++ core. When lambda = NULL, the same native correlation matrix deter-

mines the returned default lambda path. A user-supplied path must be finite, strictly positive, and non-increasing; tied values are allowed and are used unchanged. If smaller generated values cannot be certified to the solver's KKT tolerance, the function warns and returns the longest certified path prefix. A user-supplied value that cannot be certified raises an error.

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 = TRUE`, 3 plotted graphs are aligned to the same sparsity levels. The default value is `FALSE`.
`...`	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"
...	System reserved (No specific usage)

See Also

huge.select

---

plot.sim *Plot function for S3 class "sim"*

Description

Visualize the covariance matrix, the empirical correlation 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"
...	System reserved (No specific usage)

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

stockdata *Stock price of S&P 500 companies from 2003 to 2008*

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

∗ **datasets**
stockdata, 27

huge, *3*, 4, *4*, *7, 8*, *10–18*, *21*, *23*, *25*
huge-package, 3
huge.ct, 7
huge.generator, *3, 4*, *7*, 8, *25, 26*
huge.glasso, 11
huge.inference, 12
huge.mb, 14
huge.npn, *3, 4*, 15
huge.plot, *4*, *7*, 16
huge.roc, *4*, *7*, 18, *24, 25*
huge.select, *3, 4*, *6, 7*, 19, *24*, *26*
huge.tiger, *3, 4*, 21

plot.huge, 23
plot.roc, 23
plot.select, 24
plot.sim, 24
print.huge, 25
print.roc, 25
print.select, 26
print.sim, 26

rank, *16*

stockdata, 27