Package 'nethet'

March 30, 2021

Type Package

Title A bioconductor package for high-dimensional exploration of biological network heterogeneity

Version 1.22.0

Date 2020-09-27

Author Nicolas Staedler, Frank Dondelinger

Maintainer Nicolas Staedler <staedler.n@gmail.com>, Frank Dondelinger <fdondelinger.work@gmail.com>

Description Package nethet is an implementation of statistical solid methodology enabling the analysis of network heterogeneity from high-dimensional data. It combines several implementations of recent statistical innovations useful for estimation and comparison of networks in a heterogeneous, high-dimensional setting. In particular, we provide code for formal two-sample testing in Gaussian graphical models (differential network and GGM-GSA; Stadler and Mukherjee, 2013, 2014) and make a novel network-based clustering algorithm available (mixed graphical lasso, Stadler and Mukherjee, 2013).

Imports glasso, mvtnorm, GeneNet, huge, CompQuadForm, ggm, mclust, parallel, GSA, limma, multtest, ICSNP, glmnet, network, ggplot2, grDevices, graphics, stats, utils

Suggests knitr, xtable, BiocStyle, testthat

biocViews Clustering, GraphAndNetwork

VignetteBuilder knitr

License GPL-2

RoxygenNote 6.1.1

git_url https://git.bioconductor.org/packages/nethet

git_branch RELEASE_3_12

git_last_commit 09eb075

git_last_commit_date 2020-10-27

Date/Publication 2021-03-29

R topics documented:

`NetHet-package` *NetHet-package*

Description

A bioconductor package for high-dimensional exploration of biological network heterogeneity

Details

Includes: *Network-based clustering (MixGLasso) *Differential network (DiffNet) *Differential regression (DiffRegr) *Gene-set analysis based on graphical models (GGMGSA) *Plotting functions for exploring network heterogeneity

References

St\"adler, N. and Mukherjee, S. (2013). Two-Sample Testing in High-Dimensional Models. Preprint http://arxiv.org/abs/1210.4584.

`aggpval` *Meinshausen p-value aggregation*

Description

Meinshausen p-value aggregation.

Usage

```
aggpval(pval, gamma.min = 0.05)
```

Arguments

`pval`	Vector of p-values.
`gamma.min`	See inf-quantile formula of Meinshausen et al 2009 (default=0.05).

Details

Inf-quantile formula for p-value aggregation presented in Meinshausen et al 2009.

Value

Aggregated p-value.

Author(s)

n.stadler

Examples

```
pval=runif(50)
aggpval(pval)
```

bwprun_mixglasso *bwprun_mixglasso*

Description

Mixglasso with backward pruning

Usage

```
bwprun_mixglasso(x, n.comp.min = 1, n.comp.max, lambda = sqrt(2 *
  nrow(x) * log(ncol(x)))/2, pen = "glasso.parcor",
  selection.crit = "mmdl", term = 10^{     -3 }, min.compsize = 5,
  init = "kmeans.hc", my.cl = NULL, modelname.hc = "VVV",
  nstart.kmeans = 1, iter.max.kmeans = 10, reinit.out = FALSE,
  reinit.in = FALSE, mer = TRUE, del = TRUE, ...)
```

Arguments

x	Input data matrix
n.comp.min	Minimum number of components. Take n.comp.min=1 !
n.comp.max	Maximum number of components
lambda	Regularization parameter. Default=sqrt(2*n*log(p))/2
pen	Determines form of penalty: glasso.parcor (default), glasso.invcov, glasso.invcor
selection.crit	Selection criterion. Default='mmdl'
term	Termination criterion of EM algorithm. Default=10^-3
min.compsize	Stop EM if any(compsize)<min.compsize; Default=5
init	Initialization. Method used for initialization init='cl.init','r.means','random','kmeans','kmeans.hc','h Default='kmeans.hc'
my.cl	Initial cluster assignments; need to be provided if init='cl.init' (otherwise this param is ignored). Default=NULL
modelname.hc	Model class used in hc. Default="VVV"
nstart.kmeans	Number of random starts in kmeans; default=1
iter.max.kmeans	Maximal number of iteration in kmeans; default=10
reinit.out	Re-initialization if compsize<min.compsize (at the start of algorithm) ?
reinit.in	Re-initialization if compsize<min.compsize (at the bwprun-loop level of algorithm) ?
mer	Merge closest comps for initialization
del	Delete smallest comp for initialization
...	Other arguments. See mixglasso_init

Details

This function runs mixglasso with various number of mixture components: It starts with a too large number of components and iterates towards solutions with smaller number of components by initializing using previous solutions.

Value

list consisting of

selcrit	Selcrit for all models with number of components between n.comp.min and n.comp.max
res.init	Initialization for all components
comp.name	List of names of components. Indicates which states where merged/deleted during backward pruning
re.init.in	Logical vector indicating whether re-initialization was performed or not
fit.mixgl.selcrit	Results for model with optimal number of components. List see mixglasso_init

Author(s)

n.stadler

Examples

```
##generate data
set.seed(1)
n <- 1000
n.comp <- 3
p <- 10

# Create different mean vectors
Mu <- matrix(0,p,n.comp)

nonzero.mean <- split(sample(1:p),rep(1:n.comp,length=p))
for(k in 1:n.comp){
  Mu[nonzero.mean[[k]],k] <- -2/sqrt(ceiling(p/n.comp))
}

sim <- sim_mix_networks(n, p, n.comp, Mu=Mu)

##run mixglasso

fit <-  bwprun_mixglasso(sim$data,n.comp=1,n.comp.max=5,selection.crit='bic')
plot(fit$selcrit,ylab='bic',xlab='Num.Comps',type='b')
```

---

diffnet_multisplit *Differential Network*

Description

Differential Network

Usage

```
diffnet_multisplit(x1, x2, b.splits = 50, frac.split = 1/2,
  screen.meth = "screen_bic.glasso", include.mean = FALSE,
  gamma.min = 0.05, compute.evals = "est2.my.ev3",
  algorithm.mleggm = "glasso_rho0", method.compquadform = "imhof",
  acc = 1e-04, epsabs = 1e-10, epsrel = 1e-10, show.warn = FALSE,
  save.mle = FALSE, verbose = TRUE, mc.flag = FALSE,
  mc.set.seed = TRUE, mc.preschedule = TRUE,
  mc.cores = getOption("mc.cores", 2L), ...)
```

Arguments

x1	Data-matrix sample 1. You might need to center and scale your data-matrix.
x2	Data-matrix sample 1. You might need to center and scale your data-matrix.
b.splits	Number of splits (default=50).
frac.split	Fraction train-data (screening) / test-data (cleaning) (default=0.5).
screen.meth	Screening procedure. Options: 'screen_bic.glasso' (default), 'screen_cv.glasso', 'screen_shrink' (not recommended).
include.mean	Should sample specific means be included in hypothesis? Use include.mean=FALSE (default and recommended) which assumes mu1=mu2=0 and tests the hypothesis H0: Omega_1=Omega_2.
gamma.min	Tuning parameter in p-value aggregation of Meinshausen et al (2009). (Default=0.05).
compute.evals	Method to estimate the weights in the weighted-sum-of-chi2s distribution. The default and (currently) the only available option is the method 'est2.my.ev3'.
algorithm.mleggm	Algorithm to compute MLE of GGM. The algorithm 'glasso_rho' is the default and (currently) the only available option.
method.compquadform	Method to compute distribution function of weighted-sum-of-chi2s (default='imhof').
acc	See ?davies (default 1e-04).
epsabs	See ?imhof (default 1e-10).
epsrel	See ?imhof (default 1e-10).
show.warn	Should warnings be showed (default=FALSE)?
save.mle	If TRUE, MLEs (inverse covariance matrices for samples 1 and 2) are saved for all b.splits. The median aggregated inverse covariance matrix is provided in the output as 'medwi'. The default is save.mle=FALSE.
verbose	If TRUE, show output progress.
mc.flag	If TRUE use parallel execution for each b.splits via function mclapply of package parallel.
mc.set.seed	See mclapply. Default=TRUE
mc.preschedule	See mclapply. Default=TRUE
mc.cores	Number of cores to use in parallel execution. Defaults to mc.cores option if set, or 2 otherwise.
...	Additional arguments for screen.meth.

Details

Remark:

* If include.mean=FALSE, then x1 and x2 have mean zero and DiffNet tests the hypothesis H0: Omega_1=Omega_2. You might need to center x1 and x2. * If include.mean=TRUE, then DiffNet tests the hypothesis H0: mu_1=mu_2 & Omega_1=Omega_2 * However, we recommend to set include.mean=FALSE and to test equality of the means separately. * You might also want to scale x1 and x2, if you are only interested in differences due to (partial) correlations.

Value

list consisting of

`ms.pval`	p-values for all b.splits
`ss.pval`	single-split p-value
`medagg.pval`	median aggregated p-value
`meinshagg.pval`	meinshausen aggregated p-value (meinshausen et al 2009)
`teststat`	test statistics for b.splits
`weights.nulldistr`	estimated weights
`active.last`	active-sets obtained in last screening-step
`medwi`	median of inverse covariance matrices over b.splits
`sig.last`	constrained mle (covariance matrix) obtained in last cleaning-step
`wi.last`	constrained mle (inverse covariance matrix) obtained in last cleaning-step

Author(s)

n.stadler

Examples

```
############################################################
##This example illustrates the use of Differential Network##
############################################################


##set seed
set.seed(1)

##sample size and number of nodes
n <- 40
p <- 10

##specifiy sparse inverse covariance matrices
gen.net <- generate_2networks(p,graph='random',n.nz=rep(p,2),
                              n.nz.common=ceiling(p*0.8))
invcov1 <- gen.net[[1]]
invcov2 <- gen.net[[2]]
plot_2networks(invcov1,invcov2,label.pos=0,label.cex=0.7)

##get corresponding correlation matrices
cor1 <- cov2cor(solve(invcov1))
```

```
cor2 <- cov2cor(solve(invcov2))

##generate data under null hypothesis (both datasets have the same underlying
## network)
library('mvtnorm')
x1 <- rmvnorm(n,mean = rep(0,p), sigma = cor1)
x2 <- rmvnorm(n,mean = rep(0,p), sigma = cor1)

##run diffnet (under null hypothesis)
dn.null <- diffnet_multisplit(x1,x2,b.splits=1,verbose=FALSE)
dn.null$ss.pval#single-split p-value

##generate data under alternative hypothesis (datasets have different networks)
x1 <- rmvnorm(n,mean = rep(0,p), sigma = cor1)
x2 <- rmvnorm(n,mean = rep(0,p), sigma = cor2)

##run diffnet (under alternative hypothesis)
dn.altn <- diffnet_multisplit(x1,x2,b.splits=1,verbose=FALSE)
dn.altn$ss.pval#single-split p-value
dn.altn$medagg.pval#median aggregated p-value

##typically we would choose a larger number of splits
# dn.altn <- diffnet_multisplit(x1,x2,b.splits=10,verbose=FALSE)
# dn.altn$ms.pval#multi-split p-values
# dn.altn$medagg.pval#median aggregated p-value
# plot(dn.altn)#histogram of single-split p-values
```

`diffnet_singlesplit` *Differential Network for user specified data splits*

Description

Differential Network for user specified data splits

Usage

```
diffnet_singlesplit(x1, x2, split1, split2,
  screen.meth = "screen_bic.glasso", compute.evals = "est2.my.ev3",
  algorithm.mleggm = "glasso_rho0", include.mean = FALSE,
  method.compquadform = "imhof", acc = 1e-04, epsabs = 1e-10,
  epsrel = 1e-10, show.warn = FALSE, save.mle = FALSE, ...)
```

Arguments

`x1`	Data-matrix sample 1. You might need to center and scale your data-matrix.
`x2`	Data-matrix sample 2. You might need to center and scale your data-matrix.
`split1`	Samples (condition 1) used in screening step.
`split2`	Samples (condition 2) used in screening step.
`screen.meth`	Screening procedure. Options: 'screen_bic.glasso' (default), 'screen_cv.glasso', 'screen_shrink' (not recommended).
`compute.evals`	Method to estimate the weights in the weighted-sum-of-chi2s distribution. The default and (currently) the only available option is the method 'est2.my.ev3'.

`algorithm.mleggm`
: Algorithm to compute MLE of GGM. The algorithm 'glasso_rho' is the default and (currently) the only available option.

`include.mean`
: Should sample specific means be included in hypothesis? Use include.mean=FALSE (default and recommended) which assumes mu1=mu2=0 and tests the hypothesis H0: Omega_1=Omega_2.

`method.compquadform`
: Method to compute distribution function of weighted-sum-of-chi2s (default='imhof').

`acc`
: See ?davies (default 1e-04).

`epsabs`
: See ?imhof (default 1e-10).

`epsrel`
: See ?imhof (default 1e-10).

`show.warn`
: Should warnings be showed (default=FALSE)?

`save.mle`
: Should MLEs be in the output list (default=FALSE)?

`...`
: Additional arguments for screen.meth.

Details

Remark:

* If include.mean=FALSE, then x1 and x2 have mean zero and DiffNet tests the hypothesis H0: Omega_1=Omega_2. You might need to center x1 and x2. * If include.mean=TRUE, then DiffNet tests the hypothesis H0: mu_1=mu_2 & Omega_1=Omega_2 * However, we recommend to set include.mean=FALSE and to test equality of the means separately. * You might also want to scale x1 and x2, if you are only interested in differences due to (partial) correlations.

Value

list consisting of

`pval.onesided`
: p-value

`pval.twosided`
: ignore this output

`teststat`
: log-likelihood-ratio test statistic

`weights.nulldistr`
: estimated weights

`active`
: active-sets obtained in screening-step

`sig`
: constrained mle (covariance) obtained in cleaning-step

`wi`
: constrained mle (inverse covariance) obtained in cleaning-step

`mu`
: mle (mean) obtained in cleaning-step

Author(s)

n.stadler

Examples

```
##set seed
set.seed(1)

##sample size and number of nodes
n <- 40
```

```
p <- 10

##specifiy sparse inverse covariance matrices
gen.net <- generate_2networks(p,graph='random',n.nz=rep(p,2),
                              n.nz.common=ceiling(p*0.8))
invcov1 <- gen.net[[1]]
invcov2 <- gen.net[[2]]
plot_2networks(invcov1,invcov2,label.pos=0,label.cex=0.7)

##get corresponding correlation matrices
cor1 <- cov2cor(solve(invcov1))
cor2 <- cov2cor(solve(invcov2))

##generate data under alternative hypothesis
library('mvtnorm')
x1 <- rmvnorm(n,mean = rep(0,p), sigma = cor1)
x2 <- rmvnorm(n,mean = rep(0,p), sigma = cor2)

##run diffnet
split1 <- sample(1:n,20)#samples for screening (condition 1)
split2 <- sample(1:n,20)#samples for screening (condition 2)
dn <- diffnet_singlesplit(x1,x2,split1,split2)
dn$pval.onesided#p-value
```

`diffregr_multisplit` *Differential Regression (multi-split version).*

Description

Differential Regression (multi-split version).

Usage

```
diffregr_multisplit(y1, y2, x1, x2, b.splits = 50, frac.split = 1/2,
  screen.meth = "screen_cvtrunc.lasso", gamma.min = 0.05,
  compute.evals = "est2.my.ev3.diffregr",
  method.compquadform = "imhof", acc = 1e-04, epsabs = 1e-10,
  epsrel = 1e-10, show.warn = FALSE, n.perm = NULL,
  mc.flag = FALSE, mc.set.seed = TRUE, mc.preschedule = TRUE,
  mc.cores = getOption("mc.cores", 2L), ...)
```

Arguments

`y1`	Response vector condition 1.
`y2`	Response vector condition 2.
`x1`	Predictor matrix condition 1.
`x2`	Predictor matrix condition 2.
`b.splits`	Number of splits (default=50).
`frac.split`	Fraction train-data (screening) / test-data (cleaning) (default=0.5).
`screen.meth`	Screening method (default='screen_cvtrunc.lasso').
`gamma.min`	Tuning parameter in p-value aggregation of Meinshausen et al (2009) (default=0.05).

compute.evals	Method to estimate the weights in the weighted-sum-of-chi2s distribution. The default and (currently) the only available option is the method 'est2.my.ev3.diffregr'.
method.compquadform	Algorithm for computing distribution function of weighted-sum-of-chi2 (default='imhof').
acc	See ?davies (default=1e-4).
epsabs	See ?imhof (default=1e-10).
epsrel	See ?imhof (default=1e-10).
show.warn	Show warnings (default=FALSE)?
n.perm	Number of permutation for "split-perm" p-value. Default=NULL, which means that the asymptotic approximation is used.
mc.flag	If TRUE use parallel execution for each b.splits via function mclapply of package parallel.
mc.set.seed	See mclapply. Default=TRUE
mc.preschedule	See mclapply. Default=TRUE
mc.cores	Number of cores to use in parallel execution. Defaults to mc.cores option if set, or 2 otherwise.
...	Other arguments specific to screen.meth.

Details

Intercepts in regression models are assumed to be zero (mu1=mu2=0). You might need to center the input data prior to running Differential Regression.

Value

List consisting of

ms.pval	p-values for all b.splits
ss.pval	single-split p-value
medagg.pval	median aggregated p-value
meinshagg.pval	meinshausen aggregated p-value (meinshausen et al 2009)
teststat	test statistics for b.splits
weights.nulldistr	estimated weights
active.last	active-sets obtained in last screening-step
beta.last	constrained mle (regression coefficients) obtained in last cleaning-step

Author(s)

n.stadler

Examples

```
#################################################################
##This example illustrates the use of Differential Regression##
#################################################################

##set seed
set.seed(1)

## Number of predictors and sample size
p <- 100
n <- 80

## Predictor matrices
x1 <- matrix(rnorm(n*p),n,p)
x2 <- matrix(rnorm(n*p),n,p)

## Active-sets and regression coefficients
act1 <- sample(1:p,5)
act2 <- c(act1[1:3],sample(setdiff(1:p,act1),2))
beta1 <- beta2 <- rep(0,p)
beta1[act1] <- 0.5
beta2[act2] <- 0.5

## Response vectors under null-hypothesis
y1 <- x1%*%as.matrix(beta1)+rnorm(n,sd=1)
y2 <- x2%*%as.matrix(beta1)+rnorm(n,sd=1)

## Diffregr (asymptotic p-values)
fit.null <- diffregr_multisplit(y1,y2,x1,x2,b.splits=5)
fit.null$ms.pval#multi-split p-values
fit.null$medagg.pval#median aggregated p-values

## Response vectors under alternative-hypothesis
y1 <- x1%*%as.matrix(beta1)+rnorm(n,sd=1)
y2 <- x2%*%as.matrix(beta2)+rnorm(n,sd=1)

## Diffregr (asymptotic p-values)
fit.alt <- diffregr_multisplit(y1,y2,x1,x2,b.splits=5)
fit.alt$ms.pval
fit.alt$medagg.pval

## Diffregr (permutation-based p-values; 100 permutations)
fit.alt.perm <- diffregr_multisplit(y1,y2,x1,x2,b.splits=5,n.perm=100)
fit.alt.perm$ms.pval
fit.alt.perm$medagg.pval
```

`diffregr_pval` *Computation "split-asym" p-values.*

Description

Computation "split-asym"/"split-perm" p-values.

Usage

```
diffregr_pval(y1, y2, x1, x2, beta1, beta2, beta, act1, act2, act,
  compute.evals, method.compquadform, acc, epsabs, epsrel, show.warn,
  n.perm)
```

Arguments

y1	Response vector condition 1.
y2	Response vector condition 2.
x1	Predictor matrix condition 1.
x2	Predictor matrix condition 2.
beta1	Regression coefficients condition 1.
beta2	Regression coefficients condition 2.
beta	Pooled regression coefficients.
act1	Active-set condition 1.
act2	Active-set condition 2.
act	Pooled active-set.
compute.evals	Method for computation of weights.
method.compquadform	Method to compute distribution function of w-sum-of-chi2.
acc	See ?davies.
epsabs	See ?imhof.
epsrel	See ?imhof.
show.warn	Show warnings?
n.perm	Number of permutations.

Value

P-value, test statistic, estimated weights.

Author(s)

n.stadler

---

diffregr_singlesplit *Differential Regression (single-split version).*

Description

Differential Regression (single-split version).

Usage

```
diffregr_singlesplit(y1, y2, x1, x2, split1, split2,
  screen.meth = "screen_cvtrunc.lasso",
  compute.evals = "est2.my.ev3.diffregr",
  method.compquadform = "imhof", acc = 1e-04, epsabs = 1e-10,
  epsrel = 1e-10, show.warn = FALSE, n.perm = NULL, ...)
```

Arguments

y1	Response vector condition 1.
y2	Response vector condition 2.
x1	Predictor matrix condition 1.
x2	Predictor matrix condition 2.
split1	Samples condition 1 used in screening-step.
split2	Samples condition 2 used in screening-step.
screen.meth	Screening method (default='screen_cvtrunc.lasso').
compute.evals	Method to estimate the weights in the weighted-sum-of-chi2s distribution. The default and (currently) the only available option is the method 'est2.my.ev3.diffregr'.
method.compquadform	Algorithm for computing distribution function of weighted-sum-of-chi2 (default='imhof').
acc	See ?davies (default=1e-4).
epsabs	See ?imhof (default=1e-10).
epsrel	See ?imhof (default=1e-10).
show.warn	Show warnings (default=FALSE)?
n.perm	Number of permutation for "split-perm" p-value (default=NULL).
...	Other arguments specific to screen.meth.

Details

Intercepts in regression models are assumed to be zero (mu1=mu2=0). You might need to center the input data prior to running Differential Regression.

Value

List consisting of

pval.onesided	"One-sided" p-value.
pval.twosided	"Two-sided" p-value. Ignore all "*.twosided results.
teststat	2 times Log-likelihood-ratio statistics
weights.nulldistr	Estimated weights of weighted-sum-of-chi2s.
active	List of active-sets obtained in screening step.
beta	Regression coefficients (MLE) obtaind in cleaning-step.

Author(s)

n.stadler

Examples

```
##set seed
set.seed(1)

##number of predictors / sample size
p <- 100
n <- 80

##predictor matrices
x1 <- matrix(rnorm(n*p),n,p)
x2 <- matrix(rnorm(n*p),n,p)

##active-sets and regression coefficients
act1 <- sample(1:p,5)
act2 <- c(act1[1:3],sample(setdiff(1:p,act1),2))
beta1 <- beta2 <- rep(0,p)
beta1[act1] <- 0.5
beta2[act2] <- 0.5

##response vectors
y1 <- x1%*%as.matrix(beta1)+rnorm(n,sd=1)
y2 <- x2%*%as.matrix(beta2)+rnorm(n,sd=1)

##run diffregr
split1 <- sample(1:n,50)#samples for screening (condition 1)
split2 <- sample(1:n,50)#samples for screening (condition 2)
fit <- diffregr_singlesplit(y1,y2,x1,x2,split1,split2)
fit$pval.onesided#p-value
```

---

dot_plot *Create a plot showing the edges with the highest partial correlation in any cluster.*

Description

This function takes the output of het_cv_glasso or mixglasso and creates a plot of the highest scoring edges along the y axis, where, the edge in each cluster is represented by a circle whose area is proportional to the smallest mean of the two nodes that make up the edge, and the position along the y axis shows the partial correlation of the edge.

Usage

```
dot_plot(net.clustering, p.corrs.thresh = 0.25, hard.limit = 50,
  display = TRUE, node.names = rownames(net.clustering$Mu),
  group.names = sort(unique(net.clustering$comp)),
  dot.size.range = c(3, 12))
```

Arguments

net.clustering	A network clustering object as returned by het_cv_glasso or mixglasso.
p.corrs.thresh	Cutoff for the partial correlations; only edges with absolute partial correlation > p.corrs.thresh (in any cluster) will be displayed.
hard.limit	Additional hard limit on the number of edges to display. If p.corrs.thresh results in more edges than hard.limit, only hard.limit edges with the highest partial correlation are returned.
display	If TRUE, print the plot to the current output device.
node.names	Names for the nodes in the network.
group.names	Names for the clusters or groups.
dot.size.range	Graphical parameter for scaling the size of the circles (dots) representing an edge in each cluster.

Value

Returns a ggplot2 object. If display=TRUE, additionally displays the plot.

Examples

```
n = 500
p = 10
s = 0.9
n.comp = 3

# Create different mean vectors
Mu = matrix(0,p,n.comp)

# Define non-zero means in each group (non-overlapping)
nonzero.mean = split(sample(1:p),rep(1:n.comp,length=p))

# Set non-zero means to fixed value
for(k in 1:n.comp){
Mu[nonzero.mean[[k]],k] = -2/sqrt(ceiling(p/n.comp))
}

# Generate data
sim.result = sim_mix_networks(n, p, n.comp, s, Mu=Mu)
mixglasso.result = mixglasso(sim.result$data, n.comp=3)
mixglasso.clustering = mixglasso.result$models[[mixglasso.result$bic.opt]]

dot_plot(mixglasso.clustering, p.corrs.thresh=0.5)
```

---

export_network *Export networks as a CSV table.*

Description

This function takes the output of het_cv_glasso or mixglasso and exports it as a text table in CSV format, where each entry in the table records an edge in one group and its partial correlation.

Usage

```
export_network(net.clustering, file = "network_table.csv",
  node.names = rownames(net.clustering$Mu),
  group.names = sort(unique(net.clustering$comp)),
  p.corrs.thresh = 0.2, ...)
```

Arguments

net.clustering	A network clustering object as returned by screen_cv.glasso or mixglasso.
file	Filename to save the network table under.
node.names	Names for the nodes in the network. If NULL, names from net.clustering will be used.
group.names	Names for the clusters or groups. If NULL, names from net.clustering will be used (by default these are integets 1:numClusters).
p.corrs.thresh	Threshold applied to the absolute partial correlations. Edges that are below the threshold in all of the groups are not exported. Using a negative value will export all possible edges (including those with zero partial correlation).
...	Further parameters passed to write.csv.

Value

Function does not return anything.

Author(s)

Frank Dondelinger

Examples

```
n = 500
p = 10
s = 0.9
n.comp = 3

# Create different mean vectors
Mu = matrix(0,p,n.comp)

# Define non-zero means in each group (non-overlapping)
nonzero.mean = split(sample(1:p),rep(1:n.comp,length=p))

# Set non-zero means to fixed value
for(k in 1:n.comp){
Mu[nonzero.mean[[k]],k] = -2/sqrt(ceiling(p/n.comp))
}

# Generate data
sim.result = sim_mix_networks(n, p, n.comp, s, Mu=Mu)
mixglasso.result = mixglasso(sim.result$data, n.comp=3)
mixglasso.clustering = mixglasso.result$models[[mixglasso.result$bic.opt]]

## Not run:
# Save network in CSV format suitable for Cytoscape import
export_network(mixglasso.clustering, file='nethet_network.csv',
```

```
 p.corrs.thresh=0.25, quote=FALSE)

## End(Not run)
```

---

generate_2networks *Generate sparse invcov with overlap*

Description

Generate two sparse inverse covariance matrices with overlap

Usage

```
generate_2networks(p, graph = "random", n.nz = rep(p, 2),
  n.nz.common = p, n.hub = 2, n.hub.diff = 1, magn.nz.diff = 0.8,
  magn.nz.common = 0.9, magn.diag = 0, emin = 0.1, verbose = FALSE)
```

Arguments

p	number of nodes
graph	'random' or 'hub'
n.nz	number of edges per graph (only for graph='random')
n.nz.common	number of edges incommon between graphs (only for graph='random')
n.hub	number of hubs (only for graph='hub')
n.hub.diff	number of different hubs
magn.nz.diff	default=0.9
magn.nz.common	default=0.9
magn.diag	default=0
emin	default=0.1 (see ?huge.generator)
verbose	If verbose=FALSE then tracing output is disabled.

Value

Two sparse inverse covariance matrices with overlap

Examples

```
n <- 70
p <- 30

## Specifiy sparse inverse covariance matrices,
## with number of edges in common equal to ~ 0.8*p
gen.net <- generate_2networks(p,graph='random',n.nz=rep(p,2),
                              n.nz.common=ceiling(p*0.8))

invcov1 <- gen.net[[1]]
invcov2 <- gen.net[[2]]

plot_2networks(invcov1,invcov2,label.pos=0,label.cex=0.7)
```

generate_inv_cov *generate_inv_cov*

Description

Generate an inverse covariance matrix with a given sparsity and dimensionality

Usage

```
generate_inv_cov(p = 162, sparsity = 0.7)
```

Arguments

p	Dimensionality of the matrix.
sparsity	Determined the proportion of non-zero off-diagonal entries.

Details

This function generates an inverse covariance matrix, with at most (1-sparsity)*p(p-1) non-zero off-diagonal entries, where the non-zero entries are sampled from a beta distribution.

Value

A p by p positive definite inverse covariance matrix.

Examples

```
generate_inv_cov(p=162)
```

ggmgsa_multisplit *Multi-split GGMGSA (parallelized computation)*

Description

Multi-split GGMGSA (parallelized computation)

Usage

```
ggmgsa_multisplit(x1, x2, b.splits = 50, gene.sets, gene.names,
  gs.names = NULL, method.p.adjust = "fdr",
  order.adj.agg = "agg-adj", mc.flag = FALSE, mc.set.seed = TRUE,
  mc.preschedule = TRUE, mc.cores = getOption("mc.cores", 2L),
  verbose = TRUE, ...)
```

Arguments

x1	Expression matrix for condition 1 (mean zero is required).
x2	Expression matrix for condition 2 (mean zero is required).
b.splits	Number of random data splits (default=50).
gene.sets	List of gene-sets.
gene.names	Gene names. Each column in x1 (and x2) corresponds to a gene.
gs.names	Gene-set names (default=NULL).
method.p.adjust	Method for p-value adjustment (default='fdr').
order.adj.agg	Order of aggregation and adjustment of p-values. Options: 'agg-adj' (default), 'adj-agg'.
mc.flag	If TRUE use parallel execution for each b.splits via function mclapply of package parallel.
mc.set.seed	See mclapply. Default=TRUE
mc.preschedule	See mclapply. Default=TRUE
mc.cores	Number of cores to use in parallel execution. Defaults to mc.cores option if set, or 2 otherwise.
verbose	If TRUE, show output progess.
...	Other arguments (see diffnet_singlesplit).

Details

Computation can be parallelized over many data splits.

Value

List consisting of

medagg.pval	Median aggregated p-values
meinshagg.pval	Meinshausen aggregated p-values
pval	matrix of p-values before correction and adjustement, dim(pval)=(number of gene-sets)x(number of splits)
teststatmed	median aggregated test-statistic
teststatmed.bic	median aggregated bic-corrected test-statistic
teststatmed.aic	median aggregated aic-corrected test-statistic
teststat	matrix of test-statistics, dim(teststat)=(number of gene-sets)x(number of splits)
rel.edgeinter	normalized intersection of edges in condition 1 and 2
df1	degrees of freedom of GGM obtained from condition 1
df2	degrees of freedom of GGM obtained from condition 2
df12	degrees of freedom of GGM obtained from pooled data (condition 1 and 2)

Author(s)

n.stadler

Examples

```
#########################################################
##This example illustrates the use of GGMGSA         ##
#########################################################


## Generate networks
set.seed(1)
p <- 9#network with p nodes
n <- 40
hub.net <- generate_2networks(p,graph='hub',n.hub=3,n.hub.diff=1)#generate hub networks
invcov1 <- hub.net[[1]]
invcov2 <- hub.net[[2]]
plot_2networks(invcov1,invcov2,label.pos=0,label.cex=0.7)

## Generate data
library('mvtnorm')
x1 <- rmvnorm(n,mean = rep(0,p), sigma = cov2cor(solve(invcov1)))
x2 <- rmvnorm(n,mean = rep(0,p), sigma = cov2cor(solve(invcov2)))

## Run DiffNet
# fit.dn <- diffnet_multisplit(x1,x2,b.splits=2,verbose=FALSE)
# fit.dn$medagg.pval

## Identify hubs with 'gene-sets'
gene.names <- paste('G',1:p,sep='')
gsets <- split(gene.names,rep(1:3,each=3))

## Run GGM-GSA
fit.ggmgsa <- ggmgsa_multisplit(x1,x2,b.splits=2,gsets,gene.names,verbose=FALSE)
summary(fit.ggmgsa)
fit.ggmgsa$medagg.pval#median aggregated p-values
p.adjust(apply(fit.ggmgsa$pval,1,median),method='fdr')#or: first median aggregation,
                                                      #second fdr-correction
```

`ggmgsa_singlesplit` *Single-split GGMGSA*

Description

Single-split GGMGSA

Usage

```
ggmgsa_singlesplit(x1, x2, gene.sets, gene.names,
  method.p.adjust = "fdr", verbose = TRUE, ...)
```

Arguments

x1	centered (scaled) data for condition 1
x2	centered (scaled) data for condition 2
gene.sets	List of gene-sets.
gene.names	Gene names. Each column in x1 (and x2) corresponds to a gene.
method.p.adjust	Method for p-value adjustment (default='fdr').
verbose	If TRUE, show output progess.
...	Other arguments (see diffnet_singlesplit).

Value

List of results.

Author(s)

n.stadler

---

gsea.iriz *Irizarry approach for gene-set testing*

Description

Irizarry approach for gene-set testing

Usage

```
gsea.iriz(x1, x2, gene.sets, gene.names, gs.names = NULL,
  method.p.adjust = "fdr", alternative = "two-sided")
```

Arguments

x1	Expression matrix (condition 1)
x2	Expression matrix (condition 2)
gene.sets	List of gene-sets
gene.names	Gene names
gs.names	Gene-set names
method.p.adjust	Method for p-value adjustment (default='fdr')
alternative	Default='two-sided' (uses two-sided p-values).

Details

Implements the approach described in "Gene set enrichment analysis made simple" by Irizarry et al (2011). It tests for shift and/or change in scale of the distribution.

Value

List consisting of

pval.shift	p-values measuring shift
pval.scale	p-values measuring scale
pval.combined	combined p-values (minimum of pval.shift and pval.scale)

Author(s)

n.stadler

Examples

```
n <- 100
p <- 20
x1 <- matrix(rnorm(n*p),n,p)
x2 <- matrix(rnorm(n*p),n,p)
gene.names <- paste('G',1:p,sep='')
gsets <- split(gene.names,rep(1:4,each=5))
fit <- gsea.iriz(x1,x2,gsets,gene.names)
fit$pvals.combined

x2[,1:3] <- x2[,1:3]+0.5#variables 1-3 of first gene-set are upregulated
fit <- gsea.iriz(x1,x2,gsets,gene.names)
fit$pvals.combined
```

---

het_cv_glasso *Cross-validated glasso on heterogeneous dataset with grouping*

Description

Run glasso on a heterogeneous dataset to obtain networks (inverse covariance matrices) of the variables in the dataset for each pre-specified group of samples.

Usage

```
het_cv_glasso(data, grouping = rep(1, dim(data)[1]), mc.flag = FALSE,
  use.package = "huge", normalise = FALSE, verbose = FALSE, ...)
```

Arguments

data	The heterogenous network data. Needs to be a num.samples by dim.samples matrix or dataframe.
grouping	The grouping of samples; a vector of length num.samples, with num.groups unique elements.
mc.flag	Whether to use parallel processing via package mclapply to distribute the glasso estimation over different groups.
use.package	'glasso' for glasso package, or 'huge' for huge package (default)
normalise	If TRUE, normalise the columns of the data matrix before running glasso.
verbose	If TRUE, output progress.
...	Further parameters to be passed to screen_cv.glasso.

Details

This function runs the graphical lasso with cross-validation to determine the best parameter lambda for each group of samples. Note that this function defaults to using package huge (rather than package glasso) unless otherwise specified, as it tends to be more numerically stable.

Value

Returns a list with named elements 'Sig', 'SigInv', 'Mu', 'Sigma.diag', 'group.names' and 'var.names. The variables Sig and SigInv are arrays of size dim.samples by dim.samples by num.groups, where the first two dimensions contain the (inverse) covariance matrix for the network obtained by running glasso on group k. Variables Mu and Sigma.diag contain the mean and variance of the input data, and group.names and var.names contains the names for the groups and variables in the data (if specified as colnames of the input data matrix).

Examples

```
n = 100
p = 25

# Generate networks with random means and covariances.
sim.result = sim_mix_networks(n, p, n.comp=3)

test.data = sim.result$data
test.labels = sim.result$comp

# Reconstruct networks for each component
networks = het_cv_glasso(data=test.data, grouping=test.labels)
```

---

invcov2parcor *Convert inverse covariance to partial correlation*

Description

Convert inverse covariance to partial correlation

Usage

```
invcov2parcor(invcov)
```

Arguments

invcov	Inverse covariance matrix

Value

The partial correlation matrix.

Examples

```
inv.cov = generate_inv_cov(p=25)
p.corr = invcov2parcor(inv.cov)
```

invcov2parcor_array *Convert inverse covariance to partial correlation for several inverse covariance matrices collected in an array.*

Description

Convert inverse covariance to partial correlation for several inverse covariance matrices collected in an array.

Usage

```
invcov2parcor_array(invcov.array)
```

Arguments

invcov.array	Array of inverse covariance matrices, of dimension numNodes by numNodes by numComps.

Value

Array of partial correlation matrices of dimension numNodes by numNodes by numComps

Examples

```
invcov.array = sapply(1:5, function(x) generate_inv_cov(p=25), simplify='array')
p.corr = invcov2parcor_array(invcov.array)
```

logratio *Log-likelihood-ratio statistics used in DiffNet*

Description

Log-likelihood-ratio statistics used in Differential Network

Usage

```
logratio(x1, x2, x, sig1, sig2, sig, mu1, mu2, mu)
```

Arguments

x1	data-matrix sample 1
x2	data-matrix sample 2
x	pooled data-matrix
sig1	covariance sample 1
sig2	covariance sample 2
sig	pooled covariance
mu1	mean sample 1
mu2	mean sample 2
mu	pooled mean

Value

Returns a list with named elements 'twiceLR', 'sig1', 'sig2', 'sig'. 'twiceLR' is twice the log-likelihood-ratio statistic.

Author(s)

n.stadler

Examples

```
x1=matrix(rnorm(100),50,2)
x2=matrix(rnorm(100),50,2)
logratio(x1,x2,rbind(x1,x2),diag(1,2),diag(1,2),diag(1,2),c(0,0),c(0,0),c(0,0))$twiceLR
```

`mixglasso` *mixglasso*

Description

mixglasso

Usage

```
mixglasso(x, n.comp, lambda = sqrt(2 * nrow(x) * log(ncol(x)))/2,
  pen = "glasso.parcor", init = "kmeans.hc", my.cl = NULL,
  modelname.hc = "VVV", nstart.kmeans = 1, iter.max.kmeans = 10,
  term = 10^{     -3 }, min.compsize = 5, save.allfits = FALSE,
  filename = "mixglasso_fit.rda", mc.flag = FALSE,
  mc.set.seed = FALSE, mc.preschedule = FALSE,
  mc.cores = getOption("mc.cores", 2L), ...)
```

Arguments

`x`	Input data matrix
`n.comp`	Number of mixture components. If n.comp is a vector, `mixglasso` will estimate a model for each number of mixture components, and return a list of models, as well as their BIC and MMDL scores and the index of the best model according to each score.
`lambda`	Regularization parameter. Default=sqrt(2*n*log(p))/2
`pen`	Determines form of penalty: glasso.parcor (default) to penalise the partial correlation matrix, glasso.invcov to penalise the inverse covariance matrix (this corresponds to classical graphical lasso), glasso.invcor to penalise the inverse correlation matrix.
`init`	Initialization. Method used for initialization init='cl.init','r.means','random','kmeans','kmeans.hc','h Default='kmeans'
`my.cl`	Initial cluster assignments; need to be provided if init='cl.init' (otherwise this param is ignored). Default=NULL
`modelname.hc`	Model class used in hc. Default="VVV"
`nstart.kmeans`	Number of random starts in kmeans; default=1

iter.max.kmeans	Maximal number of iteration in kmeans; default=10
term	Termination criterion of EM algorithm. Default=10^-3
min.compsize	Stop EM if any(compsize)<min.compsize; Default=5
save.allfits	If TRUE, save output of mixglasso for all k's.
filename	If save.allfits is TRUE, output of mixglasso will be saved as paste(filename,_fit.mixgl_k.rd
mc.flag	If TRUE use parallel execution for each n.comp via function mclapply of package parallel.
mc.set.seed	See mclapply. Default=FALSE
mc.preschedule	See mclapply. Default=FALSE
mc.cores	Number of cores to use in parallel execution. Defaults to mc.cores option if set, or 2 otherwise.
...	Other arguments. See mixglasso_init

Details

Runs mixture of graphical lasso network clustering with one or several numbers of mixture components.

Value

A list with elements:

models	List with each element i containing an S3 object of class 'nethetclustering' that contains the result of fitting the mixture graphical lasso model with n.comps[i] components. See the documentation of mixglasso_ncomp_fixed for the description of this object.
bic	BIC for all fits.
mmdl	Minimum description length score for all fits.
comp	Component assignments for all fits.
bix.opt	Index of model with optimal BIC score.
mmdl.opt	Index of model with optimal MMDL score.

Author(s)

n.stadler

Examples

```
##########################################
##This an example of how to use MixGLasso##
##########################################

##generate data
set.seed(1)
n <- 1000
n.comp <- 3
p <- 10
```

```
# Create different mean vectors
Mu <- matrix(0,p,n.comp)

nonzero.mean <- split(sample(1:p),rep(1:n.comp,length=p))
for(k in 1:n.comp){
  Mu[nonzero.mean[[k]],k] <- -2/sqrt(ceiling(p/n.comp))
}

sim <- sim_mix_networks(n, p, n.comp, Mu=Mu)

##run mixglasso
set.seed(1)
fit1 <-  mixglasso(sim$data,n.comp=1:6)
fit1$bic
set.seed(1)
fit2 <-  mixglasso(sim$data,n.comp=6)
fit2$bic
set.seed(1)
fit3 <-  mixglasso(sim$data,n.comp=1:6,lambda=0)
set.seed(1)
fit4 <-  mixglasso(sim$data,n.comp=1:6,lambda=Inf)
#set.seed(1)
#fit5 <-  bwprun_mixglasso(sim$data,n.comp=1,n.comp.max=5,selection.crit='bic')
#plot(fit5$selcrit,ylab='bic',xlab='Num.Comps',type='b')

##compare bic
library('ggplot2')
plotting.frame <-
  data.frame(BIC= c(fit1$bic, fit3$bic, fit4$bic),
             Num.Comps=rep(1:6, 3),
             Lambda=rep(c('Default',
                          'Lambda = 0',
                          'Lambda = Inf'),
                        each=6))

p <- ggplot(plotting.frame) +
geom_line(aes(x=Num.Comps, y=BIC, colour=Lambda))

print(p)
```

---

mixglasso_init *mixglasso_init*

Description

mixglasso_init (initialization and lambda set by user)

Usage

```
mixglasso_init(x, n.comp, lambda, u.init, mix.prob.init, gamma = 0.5,
  pen = "glasso.parcor", penalize.diagonal = FALSE, term = 10^{
  -3 }, miniter = 5, maxiter = 1000, min.compsize = 5,
  show.trace = FALSE)
```

Arguments

x	Input data matrix
n.comp	Number of mixture components
lambda	Regularization parameter
u.init	Initial responsibilities
mix.prob.init	Initial component probablities
gamma	Determines form of penalty
pen	Determines form of penalty: glasso.parcor (default), glasso.invcov, glasso.invcor
penalize.diagonal	Should the diagonal of the inverse covariance matrix be penalized ? Default=FALSE (recommended)
term	Termination criterion of EM algorithm. Default=10^-3
miniter	Minimal number of EM iteration before 'stop EM if any(compsize)<min.compsize' applies. Default=5
maxiter	Maximal number of EM iteration. Default=1000
min.compsize	Stop EM if any(compsize)<min.compsize; Default=5
show.trace	Should information during execution be printed ? Default=FALSE

Details

This function runs mixglasso; requires initialization (u.init,mix.prob.init)

Value

list consisting of

mix.prob	Component probabilities
Mu	Component specific mean vectors
Sig	Component specific covariance matrices
SigInv	Component specific inverse covariance matrices
iter	Number of EM iterations
loglik	Log-likelihood
bic	-loglik+log(n)*DF/2
mmdl	-loglik+penmmdl/2
u	Component responsibilities
comp	Component assignments
compsize	Size of components
pi.comps	Component probabilities
warn	Warnings during EM algorithm

Author(s)

n.stadler

plot.diffnet *Plotting function for object of class 'diffnet'*

Description

Plotting function for object of class 'diffnet'

Usage

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

Arguments

x	object of class 'diffnet'
...	Further arguments.

Value

Histogram over multi-split p-values.

Author(s)

nicolas

plot.diffregr *Plotting function for object of class 'diffregr'*

Description

Plotting function for object of class 'diffregr'

Usage

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

Arguments

x	object of class 'diffregr'
...	Further arguments.

Value

Histogram over multi-split p-values.

Author(s)

nicolas

plot.ggmgsa *Plotting function for object of class 'ggmgmsa'*

Description

Plotting function for object of class 'ggmgsa'

Usage

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

Arguments

x	object of class 'ggmgsa'
...	Further arguments.

Value

Boxplot of single-split p-values.

Author(s)

nicolas

plot.nethetclustering *Plot networks*

Description

This function takes the output of screen_cv.glasso or mixglasso and creates a network plot using the network library.

Usage

```
## S3 method for class 'nethetclustering'
plot(x,
  node.names = rownames(net.clustering$Mu),
  group.names = sort(unique(net.clustering$comp)),
  p.corrs.thresh = 0.2, print.pdf = FALSE, pdf.filename = "networks",
  ...)
```

Arguments

x	A network clustering object as returned by screen_cv.glasso or mixglasso.
node.names	Names for the nodes in the network. If NULL, names from net.clustering will be used.
group.names	Names for the clusters or groups. If NULL, names from net.clustering will be used (by default these are integets 1:numClusters).
p.corrs.thresh	Threshold applied to the absolute partial correlations. Edges that are below the threshold in all of the groups are not displayed.
print.pdf	If TRUE, save the output as a PDF file.
pdf.filename	If print.pdf is TRUE, specifies the file name of the output PDF file.
...	Further arguments

Value

Returns NULL and prints out the networks (or saves them to pdf if print.pdf is TRUE. The networks are displayed as a series of nComps+1 plots, where in the first plot edge widths are shown according to the maximum partial correlation of the edge over all groups. The following plots show the edges for each group. Positive partial correlation edges are shown in black, negative ones in blue. If an edge is below the threshold on the absolute partial correlation, it is displayed in gray or light blue respectively.

---

plot_2networks *Plot two networks (GGMs)*

Description

Plot two networks (GGMs)

Usage

```
plot_2networks(invcov1, invcov2, node.label = paste("X", 1:nrow(invcov1),
  sep = ""), main = c("", ""), ...)
```

Arguments

invcov1	Inverse covariance matrix of GGM1.
invcov2	Inverse covariance matrix of GGM2.
node.label	Names of nodes.
main	Vector (two elements) with network names.
...	Other arguments (see plot.network).

Value

Figure with two panels (for each network).

Author(s)

nicolas

Examples

```
n <- 70
p <- 30

## Specifiy sparse inverse covariance matrices,
## with number of edges in common equal to ~ 0.8*p
gen.net <- generate_2networks(p,graph='random',n.nz=rep(p,2),
                              n.nz.common=ceiling(p*0.8))

invcov1 <- gen.net[[1]]
invcov2 <- gen.net[[2]]

plot_2networks(invcov1,invcov2,label.pos=0,label.cex=0.7)
```

---

print.nethetsummary *Print function for object of class 'nethetsummmary'*

Description

Print function for object of class 'nethetsummary'

Usage

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

Arguments

x	object of class 'nethetsummary'
...	Other arguments

Value

Function does not return anything.

Author(s)

frankd

---

scatter_plot *Create a scatterplot showing correlation between specific nodes in the network for each pre-specified group.*

Description

This function takes the output of het_cv_glasso or mixglasso and creates a plot showing the correlation between specified node pairs in the network for all groups. The subplots for each node pair are arranged in a numPairs by numGroups grid. Partial correlations associated with each node pair are also displayed.

Usage

```
scatter_plot(net.clustering, data, node.pairs, display = TRUE,
  node.names = rownames(net.clustering$Mu),
  group.names = sort(unique(net.clustering$comp)), cex = 1)
```

Arguments

net.clustering	A network clustering object as returned by het_cv_glasso or mixglasso.
data	Observed data for the nodes, a numObs by numNodes matrix. Note that nodes need to be in the same ordering as in node.names.
node.pairs	A matrix of size numPairs by 2, where each row contains a pair of nodes to display. If node.names is specified, names in node.pairs must correspond to elements of node.names.
display	If TRUE, print the plot to the current output device.
node.names	Names for the nodes in the network. If NULL, names from net.clustering will be used.
group.names	Names for the clusters or groups. If NULL, names from net.clustering will be used (by default these are integets 1:numClusters).
cex	Scale factor for text and symbols in plot.

Value

Returns a ggplot2 object. If display=TRUE, additionally displays the plot.

Examples

```
n = 500
p = 10
s = 0.9
n.comp = 3

# Create different mean vectors
Mu = matrix(0,p,n.comp)

# Define non-zero means in each group (non-overlapping)
nonzero.mean = split(sample(1:p),rep(1:n.comp,length=p))

# Set non-zero means to fixed value
for(k in 1:n.comp){
Mu[nonzero.mean[[k]],k] = -2/sqrt(ceiling(p/n.comp))
}

# Generate data
sim.result = sim_mix_networks(n, p, n.comp, s, Mu=Mu)
mixglasso.result = mixglasso(sim.result$data, n.comp=3)
mixglasso.clustering = mixglasso.result$models[[mixglasso.result$bic.opt]]

# Specify edges
node.pairs = rbind(c(1,3), c(6,9),c(7,8))

# Create scatter plots of specified edges
scatter_plot(mixglasso.clustering, data=sim.result$data,
 node.pairs=node.pairs)
```

screen_aic.glasso *AIC-tuned glasso with additional thresholding*

Description

AIC-tuned glasso with additional thresholding

Usage

```
screen_aic.glasso(x, include.mean = TRUE, length.lambda = 20,
  lambdamin.ratio = ifelse(ncol(x) > nrow(x), 0.01, 0.001),
  penalize.diagonal = FALSE, plot.it = FALSE,
  trunc.method = "linear.growth", trunc.k = 5, use.package = "huge",
  verbose = FALSE)
```

Arguments

x	The input data. Needs to be a num.samples by dim.samples matrix.
include.mean	Include mean in likelihood. TRUE / FALSE (default).
length.lambda	Length of lambda path to consider (default=20).
lambdamin.ratio	Ratio lambda.min/lambda.max.
penalize.diagonal	If TRUE apply penalization to diagonal of inverse covariance as well. (default=FALSE)
plot.it	TRUE / FALSE (default)
trunc.method	None / linear.growth (default) / sqrt.growth
trunc.k	truncation constant, number of samples per predictor (default=5)
use.package	'glasso' or 'huge' (default).
verbose	If TRUE, output la.min, la.max and la.opt (default=FALSE).

Value

Returns a list with named elements 'rho.opt', 'wi', 'wi.orig'. Variable rho.opt is the optimal (scaled) penalization parameter (rho.opt=2*la.opt/n). The variables wi and wi.orig are matrices of size dim.samples by dim.samples containing the truncated and untruncated inverse covariance matrix.

Author(s)

n.stadler

Examples

```
n=50
p=5
x=matrix(rnorm(n*p),n,p)
wihat=screen_aic.glasso(x,length.lambda=5)$wi
```

screen_bic.glasso *BIC-tuned glasso with additional thresholding*

Description

BIC-tuned glasso with additional thresholding

Usage

```
screen_bic.glasso(x, include.mean = TRUE, length.lambda = 20,
  lambdamin.ratio = ifelse(ncol(x) > nrow(x), 0.01, 0.001),
  penalize.diagonal = FALSE, plot.it = FALSE,
  trunc.method = "linear.growth", trunc.k = 5, use.package = "huge",
  verbose = FALSE)
```

Arguments

x	The input data. Needs to be a num.samples by dim.samples matrix.
include.mean	Include mean in likelihood. TRUE / FALSE (default).
length.lambda	Length of lambda path to consider (default=20).
lambdamin.ratio	Ratio lambda.min/lambda.max.
penalize.diagonal	If TRUE apply penalization to diagonal of inverse covariance as well. (default=FALSE)
plot.it	TRUE / FALSE (default)
trunc.method	None / linear.growth (default) / sqrt.growth
trunc.k	truncation constant, number of samples per predictor (default=5)
use.package	'glasso' or 'huge' (default).
verbose	If TRUE, output la.min, la.max and la.opt (default=FALSE).

Value

Returns a list with named elements 'rho.opt', 'wi', 'wi.orig', Variable rho.opt is the optimal (scaled) penalization parameter (rho.opt=2*la.opt/n). The variables wi and wi.orig are matrices of size dim.samples by dim.samples containing the truncated and untruncated inverse covariance matrix.

Author(s)

n.stadler

Examples

```
n=50
p=5
x=matrix(rnorm(n*p),n,p)
wihat=screen_bic.glasso(x,length.lambda=5)$wi
```

screen_cv.glasso *Cross-validated glasso with additional thresholding*

Description

Cross-validated glasso with additional thresholding

Usage

```
screen_cv.glasso(x, include.mean = FALSE, folds = min(10, dim(x)[1]),
  length.lambda = 20, lambdamin.ratio = ifelse(ncol(x) > nrow(x), 0.01,
  0.001), penalize.diagonal = FALSE, trunc.method = "linear.growth",
  trunc.k = 5, plot.it = FALSE, se = FALSE, use.package = "huge",
  verbose = FALSE)
```

Arguments

x	The input data. Needs to be a num.samples by dim.samples matrix.
include.mean	Include mean in likelihood. TRUE / FALSE (default).
folds	Number of folds in the cross-validation (default=10).
length.lambda	Length of lambda path to consider (default=20).
lambdamin.ratio	Ratio lambda.min/lambda.max.
penalize.diagonal	If TRUE apply penalization to diagonal of inverse covariance as well. (default=FALSE)
trunc.method	None / linear.growth (default) / sqrt.growth
trunc.k	truncation constant, number of samples per predictor (default=5)
plot.it	TRUE / FALSE (default)
se	default=FALSE.
use.package	'glasso' or 'huge' (default).
verbose	If TRUE, output la.min, la.max and la.opt (default=FALSE).

Details

Run glasso on a single dataset, using cross-validation to estimate the penalty parameter lambda. Performs additional thresholding (optionally).

Value

Returns a list with named elements 'rho.opt', 'w', 'wi', 'wi.orig', 'mu'. Variable rho.opt is the optimal (scaled) penalization parameter (rho.opt=2*la.opt/n). Variable w is the estimated covariance matrix. The variables wi and wi.orig are matrices of size dim.samples by dim.samples containing the truncated and untruncated inverse covariance matrix. Variable mu is the mean of the input data.

Author(s)

n.stadler

Examples

```
n=50
p=5
x=matrix(rnorm(n*p),n,p)
wihat=screen_cv.glasso(x,folds=2)$wi
```

`screen_cv1se.lasso` *Cross-validated Lasso screening (lambda.1se-rule)*

Description

Cross-validated Lasso screening (lambda.1se-rule)

Usage

```
screen_cv1se.lasso(x, y)
```

Arguments

x	Predictor matrix
y	Response vector

Value

Active-set

Author(s)

n.stadler

Examples

```
screen_cv1se.lasso(matrix(rnorm(5000),50,100),rnorm(50))
```

`screen_cvfix.lasso` *Cross-validated Lasso screening and upper bound on number of predictors.*

Description

Cross-validated Lasso screening and upper bound on number of predictors

Usage

```
screen_cvfix.lasso(x, y, no.predictors = 10)
```

Arguments

x	Predictor matrix.
y	Response vector.
no.predictors	Upper bound on number of active predictors,

Details

Computes Lasso coefficients (cross-validation optimal lambda). Truncates smalles coefficients to zero such that there are no more than no.predictors non-zero coefficients

Value

Active-set.

Author(s)

n.stadler

Examples

```
screen_cvfix.lasso(matrix(rnorm(5000),50,100),rnorm(50))
```

`screen_cvmin.lasso` *Cross-validation lasso screening (lambda.min-rule)*

Description

Cross-validated Lasso screening (lambda.min-rule)

Usage

```
screen_cvmin.lasso(x, y)
```

Arguments

x	Predictor matrix
y	Response vector

Value

Active-set

Author(s)

n.stadler

Examples

```
screen_cvmin.lasso(matrix(rnorm(5000),50,100),rnorm(50))
```

screen_cvsqrt.lasso *Cross-validated Lasso screening and sqrt-truncation.*

Description

Cross-validated Lasso screening and sqrt-truncation.

Usage

```
screen_cvsqrt.lasso(x, y)
```

Arguments

x	Predictor matrix.
y	Response vector.

Details

Computes Lasso coefficients (cross-validation optimal lambda). Truncates smallest coefficients to zero, such that there are no more than sqrt(n) non-zero coefficients.

Value

Active-set.

Author(s)

n.stadler

Examples

```
screen_cvsqrt.lasso(matrix(rnorm(5000),50,100),rnorm(50))
```

screen_cvtrunc.lasso *Cross-validated Lasso screening and additional truncation.*

Description

Cross-validated Lasso screening and additional truncation.

Usage

```
screen_cvtrunc.lasso(x, y, k.trunc = 5)
```

Arguments

x	Predictor matrix.
y	Response vector.
k.trunc	Truncation constant="number of samples per predictor" (default=5).

Details

Computes Lasso coefficients (cross-validation optimal lambda). Truncates smallest coefficients to zero, such that there are no more than n/k.trunc non-zero coefficients.

Value

Active-set.

Author(s)

n.stadler

Examples

```
screen_cvtrunc.lasso(matrix(rnorm(5000),50,100),rnorm(50))
```

---

sim_mix *Simulate from mixture model.*

Description

Simulate from mixture model with multi-variate Gaussian or t-distributed components.

Usage

```
sim_mix(n, n.comp, mix.prob, Mu, Sig, dist = "norm", df = 2)
```

Arguments

n	sample size
n.comp	number of mixture components ("comps")
mix.prob	mixing probablities (need to sum to 1)
Mu	matrix of component-specific mean vectors
Sig	array of component-specific covariance matrices
dist	'norm' for Gaussian components, 't' for t-distributed components
df	degrees of freedom of the t-distribution (not used for Gaussian distribution), default=2

Value

a list consisting of:

S	component assignments
X	observed data matrix

Author(s)

n.stadler

Examples

```
n.comp = 4
p = 5 # dimensionality
Mu = matrix(rep(0, p), p, n.comp)
Sigma = array(diag(p), c(p, p, n.comp))
mix.prob = rep(0.25, n.comp)

sim_mix(100, n.comp, mix.prob, Mu, Sigma)
```

`sim_mix_networks` *sim_mix_networks*

Description

Generate inverse covariances, means, mixing probabilities, and simulate data from resulting mixture model.

Usage

```
sim_mix_networks(n, p, n.comp, sparsity = 0.7, mix.prob = rep(1/n.comp,
  n.comp), Mu = NULL, Sig = NULL, ...)
```

Arguments

`n`	Number of data points to simulate.
`p`	Dimensionality of the data.
`n.comp`	Number of components of the mixture model.
`sparsity`	Determines the proportion of non-zero off-diagonal entries.
`mix.prob`	Mixture probabilities for the components; defaults to uniform distribution.
`Mu`	Means for the mixture components, a p by n.comp matrix. If NULL, sampled from a standard Gaussian.
`Sig`	Covariances for the mixture components, a p by p by n.comp array. If NULL, generated using `generate_inv_cov`.
`...`	Further arguments passed to `sim_mix`.

Details

This function generates n.comp mean vectors from a standard Gaussian and n.comp covariance matrices, with at most (1-sparsity)*p(p-1)/2 non-zero off-diagonal entries, where the non-zero entries are sampled from a beta distribution. Then it uses `sim_mix` to simulate from a mixture model with these means and covariance matrices.

Means Mu and covariance matrices Sig can also be supplied by the user.

Value

A list with components: `Mu` Means of the mixture components. `Sig` Covariances of the mixture components. `data` Simulated data, a n by p matrix. `S` Component assignments, a vector of length n.

Examples

```
# Generate dataset with 100 samples of dimensionality 30, and 4 components
test.data = sim_mix_networks(n=100, p=30, n.comp=4)
```

summary.diffnet *Summary function for object of class 'diffnet'*

Description

Summary function for object of class 'diffnet'

Usage

```
## S3 method for class 'diffnet'
summary(object, ...)
```

Arguments

object	object of class 'diffnet'
...	Other arguments.

Value

aggregated p-values

Author(s)

nicolas

summary.diffregr *Summary function for object of class 'diffregr'*

Description

Summary function for object of class 'diffregr'

Usage

```
## S3 method for class 'diffregr'
summary(object, ...)
```

Arguments

object	object of class 'diffregr
...	Other arguments

Value

aggregated p-values

Author(s)

nicolas

summary.ggmgsa *Summary function for object of class 'ggmgsa'*

Description

Summary function for object of class 'ggmgsa'

Usage

```
## S3 method for class 'ggmgsa'
summary(object, ...)
```

Arguments

object	object of class 'ggmgsa'
...	Other arguments

Value

aggregated p-values

Author(s)

nicolas

summary.nethetclustering *Summary function for object of class 'nethetclustering'*

Description

Summary function for object of class 'nethetclustering'

Usage

```
## S3 method for class 'nethetclustering'
summary(object, ...)
```

Arguments

object	object of class 'nethetclustering'
...	Other arguments

Value

Network statistics (a 'nethetsummary' object)

Author(s)

frankd

Index