

Package ‘nethet’

April 10, 2025

Type Package

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

Version 1.39.0

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 devel

git_last_commit 585f30a

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2025-04-09

Contents

| | |
|---------------------------------|----|
| NetHet-package | 4 |
| agg.pval | 5 |
| agg.score.iriz.scale | 5 |
| agg.score.iriz.shift | 6 |
| aggpval | 6 |
| aic.glasso | 7 |
| beta.mat | 8 |
| beta.mat.diffregr | 8 |
| bic.glasso | 9 |
| buildDotPlotDataFrame | 10 |
| bwprun_mixglasso | 10 |
| cv.fold | 12 |
| cv.glasso | 13 |
| diffnet_multisplit | 14 |
| diffnet_pval | 16 |
| diffnet_singlesplit | 18 |
| diffregr_multisplit | 20 |
| diffregr_pval | 22 |
| diffregr_singlesplit | 23 |
| dot_plot | 25 |
| error.bars | 27 |
| est2.my.ev2 | 27 |
| est2.my.ev2.diffregr | 28 |
| est2.my.ev3 | 29 |
| est2.my.ev3.diffregr | 30 |
| est2.ww.mat.diffregr | 31 |
| est2.ww.mat2 | 32 |
| est2.ww.mat2.diffregr | 32 |
| export_network | 33 |
| EXPStep.mix | 34 |
| func.uinit | 35 |
| generate_2networks | 36 |
| generate_inv_cov | 37 |
| getinvcov | 38 |
| ggmgsa_multisplit | 38 |
| ggmgsa_singlesplit | 40 |
| glasso.invcor | 41 |
| glasso.invcov | 42 |
| glasso.parcor | 42 |
| gsea.highdimT2 | 43 |
| gsea.iriz | 44 |
| gsea.iriz.scale | 45 |
| gsea.iriz.shift | 46 |
| gsea.t2cov | 46 |
| het_cv_glasso | 47 |
| hugepath | 48 |

| | |
|----------------------------------|----|
| inf.mat | 49 |
| invcov2parcor | 50 |
| invcov2parcor_array | 50 |
| lambda.max | 51 |
| lambdagrid_lin | 51 |
| lambdagrid_mult | 52 |
| loglik_mix | 52 |
| logratio | 53 |
| logratio.diffregr | 54 |
| make_grid | 55 |
| mcov | 55 |
| mixglasso | 56 |
| mixglasso_init | 58 |
| mixglasso_ncomp_fixed | 60 |
| mle.ggm | 61 |
| MStepGlasso | 62 |
| my.ev2.diffregr | 63 |
| my.p.adjust | 63 |
| my.ttest | 64 |
| my.ttest2 | 64 |
| mytrunc.method | 65 |
| perm.diffregr_pval | 65 |
| perm.diffregr_teststat | 66 |
| plot.diffnet | 67 |
| plot.diffregr | 67 |
| plot.gmgmsa | 68 |
| plot.nethetclustering | 68 |
| plotCV | 69 |
| plot_2networks | 70 |
| print.nethetsummary | 71 |
| q.matrix.diffregr | 71 |
| q.matrix.diffregr3 | 72 |
| q.matrix.diffregr4 | 73 |
| q.matrix3 | 73 |
| q.matrix4 | 74 |
| scatter_plot | 75 |
| screen_aic.glasso | 76 |
| screen_bic.glasso | 77 |
| screen_cv.glasso | 78 |
| screen_cv1se.lasso | 79 |
| screen_cvfix.lasso | 80 |
| screen_cvmin.lasso | 81 |
| screen_cvsqrt.lasso | 81 |
| screen_cvtrunc.lasso | 82 |
| screen_full | 83 |
| screen_shrink | 83 |
| shapiro_screen | 84 |
| sim_mix | 85 |

| | |
|------------------------------------|-----------|
| sim_mix_networks | 86 |
| sparse_conc | 87 |
| summary.diffnet | 87 |
| summary.diffregr | 88 |
| summary.gmgsa | 88 |
| summary.nethetclustering | 89 |
| sumoffdiag | 90 |
| symmkldist | 90 |
| t2cov.lr | 91 |
| t2diagcov.lr | 91 |
| test.sd | 92 |
| test.t2 | 92 |
| tr | 93 |
| twosample_single_regr | 93 |
| w.kldist | 94 |
| ww.mat | 95 |
| ww.mat.diffregr | 95 |
| ww.mat2 | 96 |
| ww.mat2.diffregr | 97 |
| Index | 98 |

 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

Stadler, N. and Mukherjee, S. (2013). Two-Sample Testing in High-Dimensional Models. Preprint <http://arxiv.org/abs/1210.4584>.

agg.pval *P-value aggregation (Meinshausen et al 2009)*

Description

P-value aggregation

Usage

```
agg.pval(gamma, pval)
```

Arguments

| | |
|-------|----------------------------|
| gamma | see Meinshausen et al 2009 |
| pval | vector of p-values |

Value

inf-quantile aggregated p-value

Author(s)

n.stadler

agg.score.iriz.scale *Irizarry aggregate score (scale)*

Description

Irizarry aggregate score (scale)

Usage

```
agg.score.iriz.scale(ttstat, geneset, gene.name)
```

Arguments

| | |
|-----------|----------|
| ttstat | no descr |
| geneset | no descr |
| gene.name | no descr |

Value

no descr

Author(s)

n.stadler

agg.score.iriz.shift *Irizarry aggregate score (shift)*

Description

Irizarry aggregate score (shift)

Usage

```
agg.score.iriz.shift(ttstat, geneset, gene.name)
```

Arguments

| | |
|-----------|----------|
| ttstat | no descr |
| geneset | no descr |
| gene.name | no descr |

Value

no descr

Author(s)

n.stadler

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)
```

aic.glasso

AIC.glasso

Description

AIC.glasso

Usage

```
aic.glasso(x, lambda, penalize.diagonal = FALSE, plot.it = TRUE,
  use.package = "huge", include.mean = FALSE)
```

Arguments

| | |
|-------------------|----------|
| x | no descr |
| lambda | no descr |
| penalize.diagonal | no descr |
| plot.it | no descr |
| use.package | no descr |
| include.mean | no descr |

Value

no descr

Author(s)

n.stadler

beta.mat *Compute beta-matrix*

Description

Compute beta-matrix

Usage

```
beta.mat(ind1, ind2, sig1, sig2, sig)
```

Arguments

| | |
|------|----------|
| ind1 | no descr |
| ind2 | no descr |
| sig1 | no descr |
| sig2 | no descr |
| sig | no descr |

Details

beta-matrix= $E[s_{ind1}(Y;sig1) s_{ind2}(Y;sig2)']$ lsig]

Value

no descr

Author(s)

n.stadler

beta.mat.diffregr *Computation beta matrix*

Description

Computation beta matrix

Usage

```
beta.mat.diffregr(ind1, ind2, beta1, beta2, beta, sig1, sig2, sig, Sig)
```


Arguments

| | |
|-------|----------|
| ind1 | no descr |
| ind2 | no descr |
| beta1 | no descr |
| beta2 | no descr |
| beta | no descr |
| sig1 | no descr |
| sig2 | no descr |
| sig | no descr |
| Sig | no descr |

Value

no descr

Author(s)

n.stadler

| | |
|------------|-------------------|
| bic.glasso | <i>BIC.glasso</i> |
|------------|-------------------|

Description

BIC.glasso

Usage

```
bic.glasso(x, lambda, penalize.diagonal = FALSE, plot.it = TRUE,  
           use.package = "huge", include.mean = FALSE)
```

Arguments

| | |
|-------------------|----------|
| x | no descr |
| lambda | no descr |
| penalize.diagonal | no descr |
| plot.it | no descr |
| use.package | no descr |
| include.mean | no descr |

Value

no descr

Author(s)

n.stadler

 buildDotPlotDataFrame *Build up dataframe for plotting dot plot with ggplot2*

Description

Internal function

Usage

```
buildDotPlotDataFrame(net.clustering, cluster.names, node.names)
```

Arguments

```
net.clustering  Clustering
cluster.names   Cluster names
node.names      Node names
```

Value

A data frame for plotting the dotPlot with ggplot2 is returned. Column P.Corr contains the partial correlations of each edge as a numeric, column Mean contains the minimum mean expression of the two proteins (e.g. if the edge is e(p1, p2), then the column contains min(mean(p1), mean(p2))), column Edge contains the name of the edge as a character string of the form "p1-p2" and column Type contains the cluster name of the cluster that the edge belongs to as a character string.

 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

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

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

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

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')
```

`cv.fold`*Make folds*

Description

Make folds

Usage`cv.fold(n, folds = 10)`**Arguments**

| | |
|-------|----------|
| n | no descr |
| folds | no descr |

Value

no descr

Author(s)

n.stadler

`cv.glasso`*Crossvalidation for GLasso*

Description

Crossvalidation for GLasso

Usage

```
cv.glasso(x, folds = 10, lambda, penalize.diagonal = FALSE,  
          plot.it = FALSE, se = TRUE, include.mean = FALSE)
```

Arguments

| | |
|--------------------------------|---------------------------|
| <code>x</code> | no descr |
| <code>folds</code> | no descr |
| <code>lambda</code> | lambda-grid (increasing!) |
| <code>penalize.diagonal</code> | no descr |
| <code>plot.it</code> | no descr |
| <code>se</code> | no descr |
| <code>include.mean</code> | no descr |

Details

! lambda-grid has to be increasing (see glassopath)

Value

no descr

Author(s)

n.stadler

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 $\mu_1=\mu_2=0$ and tests the hypothesis $H_0: \Omega_1=\Omega_2$. |
| 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 x_1 and x_2 have mean zero and DiffNet tests the hypothesis $H_0: \Omega_{11}=\Omega_{22}$. You might need to center x_1 and x_2 . * If include.mean=TRUE, then DiffNet tests the hypothesis $H_0: \mu_1=\mu_2$ & $\Omega_{11}=\Omega_{22}$ * However, we recommend to set include.mean=FALSE and to test equality of the means separately. * You might also want to scale x_1 and x_2 , 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
```


Description

P-value calculation

Usage

```
diffnet_pval(x1, x2, x, sig1, sig2, sig, mu1, mu2, mu, act1, act2, act,  
compute.ivals, include.mean, method.compquadform, acc, epsabs, epsrel,  
show.warn)
```

Arguments

| | |
|---------------------|----------|
| x1 | no descr |
| x2 | no descr |
| x | no descr |
| sig1 | no descr |
| sig2 | no descr |
| sig | no descr |
| mu1 | no descr |
| mu2 | no descr |
| mu | no descr |
| act1 | no descr |
| act2 | no descr |
| act | no descr |
| compute.ivals | no descr |
| include.mean | no descr |
| method.compquadform | no descr |
| acc | no descr |
| epsabs | no descr |
| epsrel | no descr |
| show.warn | no descr |

Value

no descr

Author(s)

n.stadler

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.eval = "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.eval | 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 $\mu_1=\mu_2=0$ and tests the hypothesis $H_0: \Omega_1=\Omega_2$. |
| 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 $H_0: \Omega_1 = \Omega_2$. You might need to center x1 and x2. * If include.mean=TRUE, then DiffNet tests the hypothesis $H_0: \mu_1 = \mu_2 \ \& \ \Omega_1 = \Omega_2$ * 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 ($\mu_1=\mu_2=0$). 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.ivals | 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 ($\mu_1=\mu_2=0$). 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.null distr | Estimated weights of weighted-sum-of-chi2s. |
| active | List of active-sets obtained in screening step. |
| beta | Regression coefficients (MLE) obtained 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.corr.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

| | |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>net.clustering</code> | A network clustering object as returned by <code>het_cv_glasso</code> or <code>mixglasso</code> . |
| <code>p.corr.thresh</code> | Cutoff for the partial correlations; only edges with absolute partial correlation $>$ <code>p.corr.thresh</code> (in any cluster) will be displayed. |
| <code>hard.limit</code> | Additional hard limit on the number of edges to display. If <code>p.corr.thresh</code> results in more edges than <code>hard.limit</code> , only <code>hard.limit</code> edges with the highest partial correlation are returned. |
| <code>display</code> | If TRUE, print the plot to the current output device. |
| <code>node.names</code> | Names for the nodes in the network. |
| <code>group.names</code> | Names for the clusters or groups. |
| <code>dot.size.range</code> | 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.corr.thresh=0.5)
```

| | |
|------------|------------------------------|
| error.bars | <i>Error bars for plotCV</i> |
|------------|------------------------------|

Description

Error bars for plotCV

Usage

```
error.bars(x, upper, lower, width = 0.02, ...)
```

Arguments

| | |
|-------|----------|
| x | no descr |
| upper | no descr |
| lower | no descr |
| width | no descr |
| ... | no descr |

Value

no descr

Author(s)

n.stadler

| | |
|-------------|------------------------------|
| est2.my.ev2 | <i>Weights of sum-w-chi2</i> |
|-------------|------------------------------|

Description

Compute weights of sum-w-chi2 (2nd order simplification)

Usage

```
est2.my.ev2(sig1, sig2, sig, act1, act2, act, include.mean = FALSE)
```

Arguments

| | |
|--------------|----------|
| sig1 | no descr |
| sig2 | no descr |
| sig | no descr |
| act1 | no descr |
| act2 | no descr |
| act | no descr |
| include.mean | no descr |

Details

*expansion of W in two directions ("dimf>dimg direction" & "dimf>dimg direction") *simplified computation of weights is obtained by assuming H0 and that $X_u \sim X_v$ holds

Value

no descr

Author(s)

n.stadler

est2.my.ev2.diffregr *Compute weights of sum-w-chi2 (2nd order simplification)*

Description

*expansion of W in two directions ("dimf>dimg direction" & "dimf>dimg direction") *simplified computation of weights is obtained by assuming H0 and that $X_u \sim X_v$ holds

Usage

est2.my.ev2.diffregr(y1, y2, x1, x2, beta1, beta2, beta, act1, act2, act)

Arguments

| | |
|-------|----------|
| y1 | no descr |
| y2 | no descr |
| x1 | no descr |
| x2 | no descr |
| beta1 | no descr |
| beta2 | no descr |
| beta | no descr |
| act1 | no descr |
| act2 | no descr |
| act | no descr |

Value

no descr

Author(s)

n.stadler

`est2.my.ev3`*Compute weights of sum-of-weighted-chi2s*

Description

Compute weights of sum-of-weighted-chi2s

Usage

```
est2.my.ev3(sig1, sig2, sig, act1, act2, act, include.mean = FALSE)
```

Arguments

| | |
|---------------------------|------------------------------------------------|
| <code>sig1</code> | MLE (covariance matrix) sample 1 |
| <code>sig2</code> | MLE (covariance matrix) sample 2 |
| <code>sig</code> | Pooled MLE (covariance matrix) |
| <code>act1</code> | Active-set sample 1 |
| <code>act2</code> | Active-set sample 2 |
| <code>act</code> | Pooled active-set |
| <code>include.mean</code> | Should the mean be included in the likelihood? |

Details

*'2nd order simplification': 1) Factor out $(1-v_i)^{(d1+d2)}$ "expansion in $\dim f > \dim g$ direction (old terminology)" 2) Factor out $(1-\mu)^{d0}$ *simplified computation of weights is obtained without further invoking H0, or assuming $X_u \sim X_v$

Value

Eigenvalues of M, respectively the weights.

Author(s)

n.stadler

est2.my.ev3.diffregr *Compute weights of sum-of-weighted-chi2s*

Description

Compute weights of sum-of-weighted-chi2s

Usage

```
est2.my.ev3.diffregr(y1, y2, x1, x2, beta1, beta2, beta, act1, act2, act)
```

Arguments

| | |
|-------|-----------------------------------------|
| y1 | Response vector sample 1. |
| y2 | Response vector sample 2. |
| x1 | Predictor matrix sample 1. |
| x2 | Predictor matrix sample 2. |
| beta1 | MLE (regression coefficients) sample 1. |
| beta2 | MLE (regression coefficients) sample 2. |
| beta | Pooled MLE (regression coefficients). |
| act1 | Active-set sample 1 |
| act2 | Active-set sample 2 |
| act | Pooled active-set |

Details

*'2nd order simplification': 1) Factor out $(1-v_i)^{(d1+d2)}$ "expansion in $\dim f > \dim g$ direction (old terminology)" 2) Factor out $(1-\mu)^{d0}$ *simplified computation of weights is obtained without further invoking H0, or assuming $X_u \sim X_v$

Value

Eigenvalues of M, respectively the weights.

Author(s)

n.stadler

est2.ww.mat.diffregr *Estimate weights*

Description

Estimate weights

Usage

```
est2.ww.mat.diffregr(y1, y2, x1, x2, beta1, beta2, beta, act1, act2, act)
```

Arguments

| | |
|-------|----------|
| y1 | no descr |
| y2 | no descr |
| x1 | no descr |
| x2 | no descr |
| beta1 | no descr |
| beta2 | no descr |
| beta | no descr |
| act1 | no descr |
| act2 | no descr |
| act | no descr |

Details

estimate W-matrix (using plug-in estimates of Beta-matrix); calculate eigenvalues(W-matrix)

Value

no descr

Author(s)

n.stadler

 est2.ww.mat2

Weights of sum-w-chi2

Description

Compute weights of sum-w-chi2 (1st order simplification)

Usage

```
est2.ww.mat2(sig1, sig2, sig, act1, act2, act, include.mean = FALSE)
```

Arguments

| | |
|--------------|----------|
| sig1 | no descr |
| sig2 | no descr |
| sig | no descr |
| act1 | no descr |
| act2 | no descr |
| act | no descr |
| include.mean | no descr |

Value

no descr

Author(s)

n.stadler

 est2.ww.mat2.diffregr *Estimate weights*

Description

Estimate weights

Usage

```
est2.ww.mat2.diffregr(y1, y2, x1, x2, beta1, beta2, beta, act1, act2, act)
```


Arguments

| | |
|-------|----------|
| y1 | no descr |
| y2 | no descr |
| x1 | no descr |
| x2 | no descr |
| beta1 | no descr |
| beta2 | no descr |
| beta | no descr |
| act1 | no descr |
| act2 | no descr |
| act | no descr |

Value

no descr

Author(s)

n.stadler

| | |
|----------------|----------------------------------------|
| export_network | <i>Export networks as a CSV table.</i> |
|----------------|----------------------------------------|

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.corr.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 integers 1:numClusters). |

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

## End(Not run)
```

EXPStep.mix

Performs EStep

Description

Performs EStep

Usage

```
EXPStep.mix(logphi, mix.prob)
```

Arguments

| | |
|----------|----------|
| logphi | no descr |
| mix.prob | no descr |

Value

list consisting of

| | |
|----|------------------|
| u | responsibilities |
| LL | loglikelihood |

Author(s)

n.stadler

func.uinit

Initialization of MixGLasso

Description

Initialization of responsibilities

Usage

```
func.uinit(x, n.comp, init = "kmeans", my.cl = NULL,
  nstart.kmeans = 1, iter.max.kmeans = 10, modelName.hc = "EII")
```

Arguments

| | |
|-----------------|------------------------------------------------------------------------------------------------------|
| x | Observed data |
| n.comp | Number of mixture components |
| init | Method used for initialization init='cl.init','r.means','random','kmeans','kmeans.hc','hc' |
| my.cl | Initial cluster assignments; need to be provided if init='cl.init' (otherwise this param is ignored) |
| nstart.kmeans | Number of random starts in kmeans; default=1 |
| iter.max.kmeans | Maximal number of iteration in kmeans; default=10 |
| modelName.hc | Model class used in hc; default='EII' |

Value

a list consisting of

u responsibilities

Author(s)

n.stadler

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 uncommon 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 $(1-\text{sparsity}) \cdot p(p-1)$ 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)
```

| | |
|-----------|---------------------------------------------------------------------------------------|
| getinvcov | <i>Generate an inverse covariance matrix with a given sparsity and dimensionality</i> |
|-----------|---------------------------------------------------------------------------------------|

Description

Generate an inverse covariance matrix with a given sparsity and dimensionality

Usage

```
getinvcov(p, s, a.diff = 5, b.diff = 5, magn.diag = 0, emin = 0.1)
```

Arguments

| | |
|-----------|--------------------|
| p | Dimensionality |
| s | Sparsity |
| a.diff | binomial parameter |
| b.diff | binomial parameter |
| magn.diag | Magnitude |
| emin | e min |

Value

Inverse covariance matrix Internal function

| | |
|-------------------|------------------------------------------------------|
| ggmgsa_multisplit | <i>Multi-split GGMGSA (parallelized computation)</i> |
|-------------------|------------------------------------------------------|

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 progress. |
| ... | 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 adjustment, $\dim(pval)=(\text{number of gene-sets})\times(\text{number of splits})$ |
| 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, $\dim(teststat)=(\text{number of gene-sets})\times(\text{number of splits})$ |
| 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.gmgsa <- gmgsa_multisplit(x1,x2,b.splits=2,gsets,gene.names,verbose=FALSE)
summary(fit.gmgsa)
fit.gmgsa$medagg.pval#median aggregated p-values
p.adjust(apply(fit.gmgsa$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 progress. |
| ... | Other arguments (see diffnet_singlesplit). |

Value

List of results.

Author(s)

n.stadler

glasso.invcor

Graphical Lasso based on inverse covariance penalty

Description

Graphical Lasso based on inverse covariance penalty

Usage

```
glasso.invcor(s, rho, penalize.diagonal, term = 10^{ -3 })
```

Arguments

| | |
|-------------------|----------|
| s | no descr |
| rho | no descr |
| penalize.diagonal | no descr |
| term | no descr |

Value

w; wi; iter

Author(s)

n.stadler

`glasso.invcov`*Graphical Lasso based on inverse correlation penalty*

Description

Graphical Lasso based on inverse correlation penalty

Usage`glasso.invcov(s, rho, penalize.diagonal, term = 10^{ -3 })`**Arguments**

| | |
|--------------------------------|----------|
| <code>s</code> | no descr |
| <code>rho</code> | no descr |
| <code>penalize.diagonal</code> | no descr |
| <code>term</code> | no descr |

Value

w; wi; iter

Author(s)

n.stadler

`glasso.parcor`*Graphical Lasso based on partial correlation penalty*

Description

Graphical Lasso based on partial correlation penalty

Usage`glasso.parcor(s, rho, penalize.diagonal, maxiter = 1000, term = 10^{ -3 }, verbose = FALSE)`

Arguments

| | |
|-------------------|------------------------------------|
| s | no descr |
| rho | no descr |
| penalize.diagonal | no descr |
| maxiter | no descr |
| term | no descr |
| verbose | set to TRUE to print out progress. |

Value

w; wi; iter

Author(s)

n.stadler

| | |
|----------------|-------------------------------|
| gsea.highdimT2 | <i>GSA based on HighdimT2</i> |
|----------------|-------------------------------|

Description

GSA based on HighdimT2

Usage

```
gsea.highdimT2(x1, x2, gene.sets, gene.names, gs.names = NULL,
  method = "test.sd", method.p.adjust = "fdr")
```

Arguments

| | |
|-----------------|----------|
| x1 | no descr |
| x2 | no descr |
| gene.sets | no descr |
| gene.names | no descr |
| gs.names | no descr |
| method | no descr |
| method.p.adjust | no descr |

Value

no descr

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

| | |
|------------------------------|------------------------------------------------|
| <code>x1</code> | Expression matrix (condition 1) |
| <code>x2</code> | Expression matrix (condition 2) |
| <code>gene.sets</code> | List of gene-sets |
| <code>gene.names</code> | Gene names |
| <code>gs.names</code> | Gene-set names |
| <code>method.p.adjust</code> | Method for p-value adjustment (default='fdr') |
| <code>alternative</code> | 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

| | |
|----------------------------|-------------------------------------------------------------------------------------|
| <code>pval.shift</code> | p-values measuring shift |
| <code>pval.scale</code> | p-values measuring scale |
| <code>pval.combined</code> | combined p-values (minimum of <code>pval.shift</code> and <code>pval.scale</code>) |

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

```

| | |
|-----------------|---------------------------------------|
| gsea.iriz.scale | <i>Irizarry approach (scale only)</i> |
|-----------------|---------------------------------------|

Description

Irizarry approach (scale only)

Usage

```

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

```

Arguments

| | |
|-----------------|----------|
| x1 | no descr |
| x2 | no descr |
| gene.sets | no descr |
| gene.names | no descr |
| gs.names | no descr |
| method.p.adjust | |
| | no descr |
| alternative | no descr |

Value

no descr

Author(s)

n.stadler

| | |
|-----------------|---------------------------------------|
| gsea.iriz.shift | <i>Irizarry approach (shift only)</i> |
|-----------------|---------------------------------------|

Description

Irizarry approach (shift only)

Usage

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

Arguments

| | |
|-----------------|----------|
| x1 | no descr |
| x2 | no descr |
| gene.sets | no descr |
| gene.names | no descr |
| gs.names | no descr |
| method.p.adjust | |
| | no descr |
| alternative | no descr |

Value

no descr

Author(s)

n.stadler

| | |
|------------|-----------------------------|
| gsea.t2cov | <i>GSA using T2cov-test</i> |
|------------|-----------------------------|

Description

GSA using T2cov-test

Usage

```
gsea.t2cov(x1, x2, gene.sets, gene.names, gs.names = NULL,
  method = "t2cov.lr", method.p.adjust = "fdr")
```

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 | method for testing equality of covariance matrices |
| method.p.adjust | method for p-value adjustment (default: 'fdr') |

Value

list of results

Author(s)

n.stadler

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)
```

hugepath

Graphical Lasso path with huge package

Description

Graphical Lasso path with huge package

Usage

```
hugepath(s, rholist, penalize.diagonal = NULL, trace = NULL)
```

Arguments

| | |
|-------------------|----------|
| s | no descr |
| rholist | no descr |
| penalize.diagonal | no descr |
| trace | no descr |

Value

no descr

Author(s)

n.stadler

inf.mat

Information Matrix of Gaussian Graphical Model

Description

Compute Information Matrix of Gaussian Graphical Model

Usage

```
inf.mat(Sig, include.mean = FALSE)
```

Arguments

Sig Sig=solve(SigInv) true covariance matrix under H0

include.mean no descr

Details

computes $E_0[s(Y;\Omega)s(Y;\Omega)']$ where $s(Y;\Omega)=(d/d\Omega) \text{LogLik}$

Value

no descr

Author(s)

n.stadler

| | |
|---------------|----------------------------------------------------------|
| invcov2parcor | <i>Convert inverse covariance to partial correlation</i> |
|---------------|----------------------------------------------------------|

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 | <i>Convert inverse covariance to partial correlation for several inverse covariance matrices collected in an array.</i> |
|---------------------|-------------------------------------------------------------------------------------------------------------------------|

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)

```

lambda.max

Lambdamax

Description

Lambdamax

Usage

```
lambda.max(x)
```

Arguments

x no descr

Value

no descr

Author(s)

n.stadler

lambdagrid_lin

Lambda-grid

Description

Lambda-grid (linear scale)

Usage

```
lambdagrid_lin(lambda.min, lambda.max, nr.gridpoints)
```

Arguments

lambda.min no descr
lambda.max no descr
nr.gridpoints no descr

Value

no descr

Author(s)

n.stadler

| | |
|-----------------|--------------------|
| lambdagrid_mult | <i>Lambda-grid</i> |
|-----------------|--------------------|

Description

Lambda-grid (log scale)

Usage

```
lambdagrid_mult(lambda.min, lambda.max, nr.gridpoints)
```

Arguments

| | |
|---------------|----------|
| lambda.min | no descr |
| lambda.max | no descr |
| nr.gridpoints | no descr |

Value

no descr

Author(s)

n.stadler

| | |
|------------|-----------------------------------------|
| loglik_mix | <i>Log-likelihood for mixture model</i> |
|------------|-----------------------------------------|

Description

Log-likelihood for mixture model

Usage

```
loglik_mix(x, mix.prob, Mu, Sig)
```

Arguments

| | |
|----------|----------|
| x | no descr |
| mix.prob | no descr |
| Mu | no descr |
| Sig | no descr |

Value

log-likelihood

Author(s)

n.stadler

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
```

logratio.diffregr *Log-likelihood ratio statistics for Differential Regression.*

Description

Log-likelihood ratio statistics for Differential Regression.

Usage

```
logratio.diffregr(y1, y2, y, xx1, xx2, xx, beta1, beta2, beta)
```

Arguments

| | |
|-------|--------------------------------------|
| y1 | Response vector condition 1. |
| y2 | Response vector condition 2. |
| y | Pooled response vector. |
| xx1 | Predictor matrix condition 1. |
| xx2 | Predictor matrix condition 2. |
| xx | Pooled predictor matrix |
| beta1 | Regression coefficients condition 1. |
| beta2 | Regression coefficients condition 2. |
| beta | Pooled regression coefficients. |

Value

2 times log-likelihood ratio statistics.

Author(s)

n.stadler

| | |
|-----------|------------------|
| make_grid | <i>Make grid</i> |
|-----------|------------------|

Description

Make grid

Usage

```
make_grid(lambda.min, lambda.max, nr.gridpoints,  
          method = "lambdagrid_mult")
```

Arguments

| | |
|---------------|----------|
| lambda.min | no descr |
| lambda.max | no descr |
| nr.gridpoints | no descr |
| method | no descr |

Value

no descr

Author(s)

n.stadler

| | |
|------|----------------------------------|
| mcov | <i>Compute covariance matrix</i> |
|------|----------------------------------|

Description

Compute covariance matrix

Usage

```
mcov(x, include.mean, covMethod = "ML")
```

Arguments

| | |
|--------------|----------|
| x | no descr |
| include.mean | no descr |
| covMethod | no descr |

Value

no descr

Author(s)

n.stadler

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','hc'. 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.rda, sep=' '). |
| 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 probabilities |
| gamma | Determines form of penalty |
| pen | Determines form of penalty: glasso.parcor (default), glasso.invcov, glasso.invcov |
| 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

 mixglasso_ncomp_fixed *mixglasso_ncomp_fixed*

Description

mixglasso_ncomp_fixed

Usage

```

mixglasso_ncomp_fixed(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, ...)

```

Arguments

| | |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------|
| x | Input data matrix |
| n.comp | Number of mixture components |
| lambda | Regularization parameter. Default=sqrt(2*n*log(p))/2 |
| pen | Determines form of penalty: glasso.parcor (default), glasso.invcov, glasso.invcor |
| init | Initialization. Method used for initialization init='cl.init', 'r.means', 'random', 'kmeans', 'kmeans.hc', 'hc'. 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 |
| ... | Other arguments. See mixglasso_init |

Details

This function runs mixglasso

Value

see return mixglasso_init. list consisting of

| | |
|----------|------------------------------|
| mix.prob | |
| Mu | |
| Sig | |
| SigInv | |
| iter | |
| loglik | |
| bic | -loglik+log(n)*DF/2 |
| mmdl | -loglik+penmmdl/2 |
| u | responsibilities |
| comp | component assignments |
| compsize | size of components |
| pi.comps | |
| warn | warnings during optimization |

Author(s)

n.stadler

mle.ggm

MLE in GGM

Description

MLE in GGM

Usage

```
mle.ggm(x, wi, algorithm = "glasso_rho0", rho = NULL, include.mean)
```

Arguments

| | |
|--------------|----------|
| x | no descr |
| wi | no descr |
| algorithm | no descr |
| rho | no descr |
| include.mean | no descr |

Value

no descr

Author(s)

n.stadler

MStepGlasso

*MStep of MixGLasso***Description**

MStep of MixGLasso

Usage

```
MStepGlasso(x, chromosome = NULL, u, v = NULL, lambda, gamma, pen,
  penalize.diagonal, equal.prob.trans = NULL, term, model = "hmm")
```

Arguments

| | |
|-------------------|----------|
| x | no descr |
| chromosome | no descr |
| u | no descr |
| v | no descr |
| lambda | no descr |
| gamma | no descr |
| pen | no descr |
| penalize.diagonal | no descr |
| equal.prob.trans | no descr |
| term | no descr |
| model | no descr |

Value

list consisting of mix.prob, Mu, Sig, SigInv

Author(s)

n.stadler

my.ev2.diffregr *Computation eigenvalues*

Description

Computation eigenvalues

Usage

```
my.ev2.diffregr(Sig, act, act1, act2)
```

Arguments

| | |
|------|----------|
| Sig | no descr |
| act | no descr |
| act1 | no descr |
| act2 | no descr |

Value

no descr

Author(s)

n.stadler

my.p.adjust *P-value adjustment*

Description

P-value adjustment

Usage

```
my.p.adjust(p, method = "fdr")
```

Arguments

| | |
|--------|------------------------------------------------|
| p | Vector of p-values. |
| method | Method for p-value adjustment (default='fdr'). |

Value

Vector of adjusted p-values.

Author(s)

n.stadler

`my.ttest`*T-test*

Description

T-test (equal variances)

Usage`my.ttest(x1, x2)`**Arguments**

x1 no descr

x2 no descr

Value

no descr

Author(s)

n.stadler

`my.ttest2`*T-test*

Description

T-test (unequal variances)

Usage`my.ttest2(x1, x2)`**Arguments**

x1 no descr

x2 no descr

Value

no descr

Author(s)

n.stadler

mytrunc.method *Additional thresholding*

Description

Additional thresholding

Usage

```
mytrunc.method(n, wi, method = "linear.growth", trunc.k = 5)
```

Arguments

| | |
|---------|----------|
| n | no descr |
| wi | no descr |
| method | no descr |
| trunc.k | no descr |

Value

no descr

Author(s)

n.stadler

perm.diffregr_pval *Computation "split-perm" p-value.*

Description

Computation "split-perm" p-value.

Usage

```
perm.diffregr_pval(y1, y2, x1, x2, act1, act2, act, n.perm)
```

Arguments

| | |
|--------|-------------------------------|
| y1 | Response vector condition 1. |
| y2 | Response vector condition 2. |
| x1 | Predictor matrix condition 1. |
| x2 | Predictor matrix condition 2. |
| act1 | Active-set condition 1. |
| act2 | Active-set condition 2. |
| act | Pooled active-set. |
| n.perm | Number of permutations. |

Value

Permutation based p-value.

Author(s)

n.stadler

perm.diffregr_teststat

Auxiliary function for computation of "split-perm" p-value.

Description

Auxiliary function for computation of "split-perm" p-value.

Usage

```
perm.diffregr_teststat(y1, y2, y12, x1, x2, x12)
```

Arguments

| | |
|-----|-------------------------------|
| y1 | Response vector condition 1. |
| y2 | Response vector condition 2. |
| y12 | Pooled response vector. |
| x1 | Predictor matrix condition 1. |
| x2 | Predictor matrix condition 2. |
| x12 | Pooled predictor matrix |

Value

Test statistic (log-likelihood-ratio statistic).

Author(s)

n.stadler

| | |
|--------------|--------------------------------------------------------|
| plot.diffnet | <i>Plotting function for object of class 'diffnet'</i> |
|--------------|--------------------------------------------------------|

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 | <i>Plotting function for object of class 'diffregr'</i> |
|---------------|---------------------------------------------------------|

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.gmgmsa`*Plotting function for object of class 'ggmgmsa'*

Description

Plotting function for object of class 'ggmgmsa'

Usage

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

Arguments

| | |
|------------------|---------------------------|
| <code>x</code> | object of class 'ggmgmsa' |
| <code>...</code> | Further arguments. |

Value

Boxplot of single-split p-values.

Author(s)

nicolas

`plot.nethetclustering` *Plot networks*

DescriptionThis 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.corr.thresh = 0.2, print.pdf = FALSE, pdf.filename = "networks",  
  ...)
```

Arguments

| | |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| x | A network clustering object as returned by <code>screen_cv.glasso</code> or <code>mixglasso</code> . |
| node.names | Names for the nodes in the network. If NULL, names from <code>net.clustering</code> will be used. |
| group.names | Names for the clusters or groups. If NULL, names from <code>net.clustering</code> will be used (by default these are integers 1:numClusters). |
| p.corr.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 <code>print.pdf</code> 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.

| | |
|--------|---------------|
| plotCV | <i>plotCV</i> |
|--------|---------------|

Description

plotCV

Usage

```
plotCV(lambda, cv, cv.error, se = TRUE, type = "b", ...)
```

Arguments

| | |
|----------|----------|
| lambda | no descr |
| cv | no descr |
| cv.error | no descr |
| se | no descr |
| type | no descr |
| ... | no descr |

Value

no descr

Author(s)

n.stadler

| | |
|----------------|---------------------------------|
| plot_2networks | <i>Plot two networks (GGMs)</i> |
|----------------|---------------------------------|

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 'nethetsummary'*

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

q.matrix.diffregr *Computation Q matrix*

Description

Computation Q matrix

Usage

```
q.matrix.diffregr(Sig, a, b, s)
```

Arguments

| | |
|-----|----------|
| Sig | no descr |
| a | no descr |
| b | no descr |
| s | no descr |

Value

no descr

Author(s)

n.stadler

`q.matrix.diffregr3` *Computation Q matrix*

Description

Computation Q matrix

Usage

```
q.matrix.diffregr3(beta.a, beta.b, beta, sig.a, sig.b, sig, Sig, act.a,  
  act.b, ss)
```

Arguments

| | |
|--------|----------|
| beta.a | no descr |
| beta.b | no descr |
| beta | no descr |
| sig.a | no descr |
| sig.b | no descr |
| sig | no descr |
| Sig | no descr |
| act.a | no descr |
| act.b | no descr |
| ss | no descr |

Value

no descr

Author(s)

n.stadler

q.matrix.diffregr4 *Computation Q matrix*

Description

Computation Q matrix

Usage

```
q.matrix.diffregr4(b.mat, act.a, act.b, ss)
```

Arguments

| | |
|-------|----------|
| b.mat | no descr |
| act.a | no descr |
| act.b | no descr |
| ss | no descr |

Value

no descr

Author(s)

n.stadler

q.matrix3 *Compute Q-matrix*

Description

Compute Q-matrix

Usage

```
q.matrix3(sig, sig.a, sig.b, act.a, act.b, ss)
```

Arguments

| | |
|-------|----------|
| sig | no descr |
| sig.a | no descr |
| sig.b | no descr |
| act.a | no descr |
| act.b | no descr |
| ss | no descr |

Value

no descr

Author(s)

n.stadler

q.matrix4

q.matrix4

Description

q.matrix4

Usage

q.matrix4(b.mat, act.a, act.b, ss)

Arguments

b.mat no descr

act.a no descr

act.b no descr

ss no descr

Value

no descr

Author(s)

n.stadler

| | |
|--------------|---------------------------------------------------------------------------------------------------------------------|
| scatter_plot | <i>Create a scatterplot showing correlation between specific nodes in the network for each pre-specified group.</i> |
|--------------|---------------------------------------------------------------------------------------------------------------------|

Description

This function takes the output of `het_cv_lasso` 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

| | |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>net.clustering</code> | A network clustering object as returned by <code>het_cv_lasso</code> or <code>mixglasso</code> . |
| <code>data</code> | Observed data for the nodes, a <code>numObs</code> by <code>numNodes</code> matrix. Note that nodes need to be in the same ordering as in <code>node.names</code> . |
| <code>node.pairs</code> | A matrix of size <code>numPairs</code> by 2, where each row contains a pair of nodes to display. If <code>node.names</code> is specified, names in <code>node.pairs</code> must correspond to elements of <code>node.names</code> . |
| <code>display</code> | If TRUE, print the plot to the current output device. |
| <code>node.names</code> | Names for the nodes in the network. If NULL, names from <code>net.clustering</code> will be used. |
| <code>group.names</code> | Names for the clusters or groups. If NULL, names from <code>net.clustering</code> will be used (by default these are integers <code>1:numClusters</code>). |
| <code>cex</code> | 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 ($\text{rho.opt}=2*\text{la.opt}/n$). 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 $\text{lambda.min}/\text{lambda.max}$.
 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 ($\text{rho.opt}=2*\text{la.opt}/n$). 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 ($\text{rho.opt}=2*\text{la.opt}/n$). 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.lse-rule)*

Description

Cross-validated Lasso screening (lambda.lse-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 | <i>Cross-validated Lasso screening and upper bound on number of predictors.</i> |
|--------------------|---------------------------------------------------------------------------------|

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 smallest 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))
```

| | |
|-------------|--------------------|
| screen_full | <i>Screen_full</i> |
|-------------|--------------------|

Description

Screen_full

Usage

```
screen_full(x, include.mean = NULL, length.lambda = NULL,
            trunc.method = NULL, trunc.k = NULL)
```

Arguments

| | |
|---------------|----------|
| x | no descr |
| include.mean | no descr |
| length.lambda | no descr |
| trunc.method | no descr |
| trunc.k | no descr |

Value

no descr

Author(s)

n.stadler

| | |
|---------------|-------------------------------------------------------------------|
| screen_shrink | <i>Shrinkage approach for estimating Gaussian graphical model</i> |
|---------------|-------------------------------------------------------------------|

Description

Shrinkage approach for estimating Gaussian graphical model

Usage

```
screen_shrink(x, include.mean = NULL, trunc.method = "linear.growth",
              trunc.k = 5)
```

Arguments

| | |
|--------------|------------------------------------------------------------------|
| x | The input data. Needs to be a num.samples by dim.samples matrix. |
| include.mean | Include mean in likelihood. TRUE / FALSE (default). |
| trunc.method | None / linear.growth (default) / sqrt.growth |
| trunc.k | truncation constant, number of samples per predictor (default=5) |

Value

Returns a list with named elements 'rho.opt', 'wi', 'wi.orig'. Variable rho.opt=NULL (no tuning parameter involved). 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

| | |
|----------------|----------------------------------|
| shapiro_screen | <i>Filter "non-normal" genes</i> |
|----------------|----------------------------------|

Description

Filter "non-normal" genes

Usage

```
shapiro_screen(x1, x2, sign.level = 0.001)
```

Arguments

| | |
|------------|--------------------------------------------------------------|
| x1 | expression matrix (condition 1) |
| x2 | expression matrix (condition 2) |
| sign.level | sign.level in Shapiro-Wilk tests (default: sign.level=0.001) |

Details

Discarding genes which have Shapiro-Wilk p-value (corrected for multiplicity) smaller than sign.level in either of the two conditions. We used sign.level=0.001 in the GGMGSA paper.

Value

list consisting of

| | |
|---------|-------------------------------------------------|
| x1.filt | expression matrix (condition 1) after filtering |
| x2.filt | expression matrix (condition 2) after filtering |

Author(s)

n.stadler

| | |
|---------|-------------------------------------|
| sim_mix | <i>Simulate from mixture model.</i> |
|---------|-------------------------------------|

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 probabilities (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 $(1-\text{sparsity}) * p(p-1)/2$ 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)
```

| | |
|-------------|-----------------------------------------------------|
| sparse_conc | <i>Generates sparse inverse covariance matrices</i> |
|-------------|-----------------------------------------------------|

Description

Generates sparse inverse covariance matrices

Usage

```
sparse_conc(p, K, s, s.common, magn.nz = 0.5, scale.parcor = TRUE)
```

Arguments

| | |
|--------------|----------------------------------------------------------------------------------|
| p | Dimensionality of inverse covariance matrix |
| K | Number of inverse covariance matrices |
| s | Number of non-zero entries per inverse covariance matrix |
| s.common | Number of non-zero entries shared across different inverse covariance matrices |
| magn.nz | Magnitude of non-zero elements |
| scale.parcor | Should SigInv be scaled to have diagonal equal one, <code>siginv=parcor</code> ? |

Value

SigInv: list of inverse covariance matrices

Author(s)

n.stadler

| | |
|-----------------|-------------------------------------------------------|
| summary.diffnet | <i>Summary function for object of class 'diffnet'</i> |
|-----------------|-------------------------------------------------------|

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.gmgmsa *Summary function for object of class 'gmgmsa'*

Description

Summary function for object of class 'gmgmsa'

Usage

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


Arguments

object object of class 'ggmsa'
... 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

| | |
|------------|---------------------------------------------|
| sumoffdiag | <i>Sum of non-diag elements of a matrix</i> |
|------------|---------------------------------------------|

Description

Sum of non-diag elements of a matrix

Usage

sumoffdiag(m)

Arguments

| | |
|---|----------|
| m | no descr |
|---|----------|

Value

Sum of non-diag elements

Author(s)

n.stadler

| | |
|------------|-----------------------------------------------------|
| symmkldist | <i>Compute symmetric kull-back leibler distance</i> |
|------------|-----------------------------------------------------|

Description

Compute symmetric kull-back leibler distance

Usage

symmkldist(mu1, mu2, sig1, sig2)

Arguments

| | |
|------|----------|
| mu1 | no descr |
| mu2 | no descr |
| sig1 | no descr |
| sig2 | no descr |

Value

symmetric kull-back leibler distance

Author(s)

n.stadler

| | |
|----------|----------------------------------------|
| t2cov.lr | <i>Classical likelihood-ratio test</i> |
|----------|----------------------------------------|

Description

Classical likelihood-ratio test (equality of covariance matrices)

Usage

```
t2cov.lr(x1, x2, include.mean = FALSE)
```

Arguments

| | |
|--------------|----------|
| x1 | no descr |
| x2 | no descr |
| include.mean | no descr |

Value

no descr

Author(s)

n.stadler

| | |
|--------------|--------------------------------------------------|
| t2diagcov.lr | <i>Diagonal-restricted likelihood-ratio test</i> |
|--------------|--------------------------------------------------|

Description

Diagonal-restricted likelihood-ratio test

Usage

```
t2diagcov.lr(x1, x2, include.mean = FALSE)
```

Arguments

| | |
|--------------|----------|
| x1 | no descr |
| x2 | no descr |
| include.mean | no descr |

Value

no descr

Author(s)

n.stadler

test.sd*High-Dim Two-Sample Test (Srivastava, 2006)*

Description

High-Dim Two-Sample Test (Srivastava, 2006)

Usage

test.sd(x1, x2)

Arguments

x1 no descr

x2 no descr

Value

no descr

Author(s)

n.stadler

test.t2*HotellingsT2*

Description

HotellingsT2

Usage

test.t2(x1, x2)

Arguments

x1 no descr

x2 no descr

Value

no descr

Author(s)

n.stadler

tr *Compute trace of matrix*

Description

Compute trace of matrix

Usage

tr(m)

Arguments

m no descr

Value

trace of matrix

Author(s)

n.stadler

twosample_single_regr *old single-split function for diffregr*

Description

Old single-split function for diffregr

Usage

```
twosample_single_regr(y1, y2, x1, x2, n.screen.pop1 = 100,  
  n.screen.pop2 = 100, screen.meth = "screen_cvmin.lasso",  
  compute.evals = "est2.my.ev3.diffregr")
```

Arguments

| | |
|---------------|----------|
| y1 | no descr |
| y2 | no descr |
| x1 | no descr |
| x2 | no descr |
| n.screen.pop1 | no descr |
| n.screen.pop2 | no descr |
| screen.meth | no descr |
| compute.eval | no descr |

Value

no descr

Author(s)

n.stadler

w.kldist

Distance between comps based on symm. kl-distance

Description

Distance between comps based on symm. kl-distance

Usage

w.kldist(Mu, Sig)

Arguments

| | |
|-----|----------|
| Mu | no descr |
| Sig | no descr |

Value

list consisting of

- comp.kldist
- min.comp.kldist

Author(s)

n.stadler

ww.mat *Weight-matrix and eigenvalues*

Description

Calculates weight-matrix and eigenvalues

Usage

```
ww.mat(imat, act, act1, act2)
```

Arguments

| | |
|------|----------|
| imat | no descr |
| act | I_uv |
| act1 | I_u |
| act2 | I_v |

Details

calculation based on true information matrix

Value

no descr

Author(s)

n.stadler

ww.mat.diffregr *Computation M matrix and eigenvalues*

Description

Computation M matrix and eigenvalues

Usage

```
ww.mat.diffregr(Sig, act, act1, act2)
```

Arguments

| | |
|------|----------|
| Sig | no descr |
| act | no descr |
| act1 | no descr |
| act2 | no descr |

Value

no descr

Author(s)

n.stadler

ww.mat2

Calculates eigenvalues of weight-matrix (using 1st order simplification)

Description

Calculates eigenvalues of weight-matrix (using 1st order simplification)

Usage

```
ww.mat2(imat, act, act1, act2)
```

Arguments

| | |
|------|----------|
| imat | no descr |
| act | I_uv |
| act1 | I_u |
| act2 | I_v |

Details

calculation based on true information matrix

Value

no descr

Author(s)

n.stadler

ww.mat2.diffregr *Computation M matrix and eigenvalues*

Description

Computation M matrix and eigenvalues

Usage

ww.mat2.diffregr(Sig, act, act1, act2)

Arguments

| | |
|------|----------|
| Sig | no descr |
| act | no descr |
| act1 | no descr |
| act2 | no descr |

Value

no descr

Author(s)

n.stadler

Index

- * **Removed**
 - screen_shrink, 83
- * **examples**
 - screen_shrink, 83
- * **export**
 - screen_shrink, 83
- * **fixed.**
 - screen_shrink, 83
- * **internal**
 - agg.pval, 5
 - agg.score.iriz.scale, 5
 - agg.score.iriz.shift, 6
 - aic.glasso, 7
 - beta.mat, 8
 - beta.mat.diffregr, 8
 - bic.glasso, 9
 - buildDotPlotDataFrame, 10
 - cv.fold, 12
 - cv.glasso, 13
 - diffnet_pval, 16
 - error.bars, 27
 - est2.my.ev2, 27
 - est2.my.ev2.diffregr, 28
 - est2.my.ev3, 29
 - est2.my.ev3.diffregr, 30
 - est2.ww.mat.diffregr, 31
 - est2.ww.mat2, 32
 - est2.ww.mat2.diffregr, 32
 - EXPStep.mix, 34
 - func.uinit, 35
 - getinvcov, 38
 - glasso.invcov, 41
 - glasso.invcov, 42
 - glasso.parcov, 42
 - gsea.highdimT2, 43
 - gsea.iriz.scale, 45
 - gsea.iriz.shift, 46
 - gsea.t2cov, 46
 - hugepath, 48
 - inf.mat, 49
 - lambda.max, 51
 - lambdagrid_lin, 51
 - lambdagrid_mult, 52
 - loglik_mix, 52
 - logratio.diffregr, 54
 - make_grid, 55
 - mcov, 55
 - mixglasso_ncomp_fixed, 60
 - mle.ggm, 61
 - MStepGlasso, 62
 - my.ev2.diffregr, 63
 - my.p.adjust, 63
 - my.ttest, 64
 - my.ttest2, 64
 - mytrunc.method, 65
 - perm.diffregr_pval, 65
 - perm.diffregr_teststat, 66
 - plotCV, 69
 - q.matrix.diffregr, 71
 - q.matrix.diffregr3, 72
 - q.matrix.diffregr4, 73
 - q.matrix3, 73
 - q.matrix4, 74
 - screen_full, 83
 - screen_shrink, 83
 - shapiro_screen, 84
 - sparse_conc, 87
 - sumoffdiag, 90
 - symmkldist, 90
 - t2cov.lr, 91
 - t2diagcov.lr, 91
 - test.sd, 92
 - test.t2, 92
 - tr, 93
 - twosample_single_regr, 93
 - w.kldist, 94
 - ww.mat, 95
 - ww.mat.diffregr, 95

- ww.mat2, 96
 - ww.mat2.diffregr, 97
 - * **is**
 - screen_shrink, 83
 - * **n=50**
 - screen_shrink, 83
 - * **p=5**
 - screen_shrink, 83
 - * **package**
 - screen_shrink, 83
 - * **parcor**
 - screen_shrink, 83
 - * **until**
 - screen_shrink, 83
 - * **wihat=screen_shrink(x)\$wi**
 - screen_shrink, 83
 - * **x=matrix(rnorm(n*p),n,p)**
 - screen_shrink, 83
- agg.pval, 5
- agg.score.iriz.scale, 5
- agg.score.iriz.shift, 6
- aggpval, 6
- aic.glasso, 7
- beta.mat, 8
- beta.mat.diffregr, 8
- bic.glasso, 9
- buildDotPlotDataFrame, 10
- bwprun_mixglasso, 10
- cv.fold, 12
- cv.glasso, 13
- diffnet_multisplit, 14
- diffnet_pval, 16
- diffnet_singlesplit, 18
- diffregr_multisplit, 20
- diffregr_pval, 22
- diffregr_singlesplit, 23
- dot_plot, 25
- error.bars, 27
- est2.my.ev2, 27
- est2.my.ev2.diffregr, 28
- est2.my.ev3, 29
- est2.my.ev3.diffregr, 30
- est2.ww.mat.diffregr, 31
- est2.ww.mat2, 32
- est2.ww.mat2.diffregr, 32
- export_network, 33
- EXPStep.mix, 34
- func.uinit, 35
- generate_2networks, 36
- generate_inv_cov, 37, 86
- getinvcov, 38
- ggmgsa_multisplit, 38
- ggmgsa_singlesplit, 40
- glasso.invcov, 41
- glasso.invcov, 42
- glasso.parcor, 42
- gsea.highdimT2, 43
- gsea.iriz, 44
- gsea.iriz.scale, 45
- gsea.iriz.shift, 46
- gsea.t2cov, 46
- het_cv_glasso, 25, 26, 33, 47, 75
- hugepath, 48
- inf.mat, 49
- invcov2parcor, 50
- invcov2parcor_array, 50
- lambda.max, 51
- lambdagrid_lin, 51
- lambdagrid_mult, 52
- loglik_mix, 52
- logratio, 53
- logratio.diffregr, 54
- make_grid, 55
- mcov, 55
- mixglasso, 25, 26, 33, 56, 68, 69, 75
- mixglasso_init, 58
- mixglasso_ncomp_fixed, 60
- mle.ggm, 61
- MStepGlasso, 62
- my.ev2.diffregr, 63
- my.p.adjust, 63
- my.ttest, 64
- my.ttest2, 64
- mytrunc.method, 65
- NetHet-package, 4
- perm.diffregr_pval, 65

perm.diffregr_teststat, 66
plot.diffnet, 67
plot.diffregr, 67
plot.gmgsa, 68
plot.nethetclustering, 68
plot_2networks, 70
plotCV, 69
print.nethetsummary, 71

q.matrix.diffregr, 71
q.matrix.diffregr3, 72
q.matrix.diffregr4, 73
q.matrix3, 73
q.matrix4, 74

scatter_plot, 75
screen_aic.glasso, 76
screen_bic.glasso, 77
screen_cv.glasso, 33, 68, 69, 78
screen_cv1se.lasso, 79
screen_cvfix.lasso, 80
screen_cvmin.lasso, 81
screen_cvsqrt.lasso, 81
screen_cvtrunc.lasso, 82
screen_full, 83
screen_shrink, 83
shapiro_screen, 84
sim_mix, 85, 86
sim_mix_networks, 86
sparse_conc, 87
summary.diffnet, 87
summary.diffregr, 88
summary.gmgsa, 88
summary.nethetclustering, 89
sumoffdiag, 90
symmklldist, 90

t2cov.lr, 91
t2diagcov.lr, 91
test.sd, 92
test.t2, 92
tr, 93
twosample_single_regr, 93

w.kldist, 94
write.csv, 34
ww.mat, 95
ww.mat.diffregr, 95
ww.mat2, 96
ww.mat2.diffregr, 97