Package ‘DGM’

May 18, 2021

Version 1.7.3
Date 2021-05-17
Title Dynamic Graphical Models
Author Simon Schwab <schw4b@gmail.com>, Ruth Harbord <r.harbord@warwick.ac.uk>, Lilia Costa <liliacosta@ufba.br>, Thomas Nichols <t.e.nichols@warwick.ac.uk>
Maintainer Simon Schwab <schw4b@gmail.com>
Depends R (>= 3.2.0)
Imports Rcpp (>= 0.11.0), data.table (>= 1.10.0), reshape2 (>= 1.4.2), ggplot2 (>= 2.2.1), coin (>= 1.2)
LinkingTo Rcpp, RcppArmadillo
Suggests testthat
Description Dynamic graphical models for multivariate time series data to estimate directed dynamic networks in functional magnetic resonance imaging (fMRI), see Schwab et al. (2017) <doi:10.1101/198887>.
License GPL-3
URL https://github.com/schw4b/DGM
BugReports https://github.com/schw4b/DGM/issues
RoxygenNote 7.1.1
Encoding UTF-8
NeedsCompilation yes
Repository CRAN
Date/Publication 2021-05-18 09:50:14 UTC

R topics documented:

  - binom.nettest .......................................................... 2
  - center .................................................................. 3
  - cor2adj .................................................................. 4
  - corTs ..................................................................... 4
Performes a binomial test with FDR correction for network edge occurrence.

Description

Performes a binomial test with FDR correction for network edge occurrence.
Usage

\texttt{binom.nettest(adj, alter = "two.sided", fdr = 0.05)}

Arguments

\texttt{adj} \hspace{1cm} \text{adjacency matrix, nodes x nodes x subj, or nodes x nodes x runs x subj.}

\texttt{alter} \hspace{1cm} \text{type of binomial test, "two.sided" (default), "less", or "greater"}

\texttt{fdr} \hspace{1cm} \text{false discovery rate (FDR) control, default is 0.05.}

Value

store list with results.

Examples

\begin{verbatim}
# Generate some sample binary 5-node network structures for N=20, then perform
# significance testing.
N=20
x = rmdiag(array(rbinom(n=5*5*N, size=1, prob=0.10), dim=c(5,5,N)))
x[1,2,2:N]=1; x[2,3,seq(1,N,2)]=1  # add some consistent edges
A = apply(x, c(1,2), mean)
l = binom.nettest(x)
\end{verbatim}

---

\textbf{center}

\textit{Mean centers timeseries in a 2D array timeseries x nodes, i.e. each timeseries of each node has mean of zero.}

Description

Mean centers timeseries in a 2D array timeseries x nodes, i.e. each timeseries of each node has mean of zero.

Usage

\texttt{center(X)}

Arguments

\texttt{X} \hspace{1cm} \text{2D array with dimensions timeseries x nodes.}

Value

\texttt{M 2D array.}
Examples

```r
data("utestdata")
myts=center(myts)
```

---

**cor2adj**

*Threshold correlation matrix to match a given number of edges.*

**Description**

Threshold correlation matrix to match a given number of edges.

**Usage**

```r
cor2adj(R, n)
```

**Arguments**

- `R`: correlation matrix.
- `n`: number of edges.

**Value**

A thresholded matrix.

---

**corTs**

*Mean correlation of time series across subjects.*

**Description**

Mean correlation of time series across subjects.

**Usage**

```r
corTs(ts)
```

**Arguments**

- `ts`: a 3D time series time series x nodes x subjects.

**Value**

M correlation matrix.
**dgm.group**

**Examples**

# create some sample data with 200 samples, # 5 nodes, and 2 subjects

```r
ts = array(rnorm(200*5*2), dim=c(200,5,2))
M = corTs(ts)
```

**Description**

A group is a list containing restructured data from subjects for easier group analysis.

**Usage**

```r
dgm.group(subj)
```

**Arguments**

`subj` a list of subjects.

**Value**

`group` a list.

**Examples**

# create some sample data with 200 samples, # 3 nodes, and 2 subjects

```r
ts = array(rnorm(200*3*2), dim=c(200,3,2))
mysubs=list()
mysubs[[1]]=subject(ts[,,1])
mysubs[[2]]=subject(ts[,,2])
g=dgm.group(mysubs)
```
diag.delta

Quick diagnostics on delta.

Usage

diag.delta(path, id, nodes)

Arguments

path path to results files.
id subject identifier.
nodes number of nodes.

Value

x array node model’s delta

dlm.lpl

Calculate the log predictive likelihood for a specified set of parents and a fixed delta.

Description

Calculate the log predictive likelihood for a specified set of parents and a fixed delta.

Usage

dlm.lpl(Yt, Ft, delta, priors = priors.spec())

Arguments

Yt the vector of observed time series, length T.
Ft the matrix of covariates, dim = number of thetas (p) x number of time points (T), usually a row of 1s to represent an intercept and the time series of the parent nodes.
delta discount factor (scalar).
priors list with prior hyperparameters.
value

t
the vector or matrix of the posterior mean (location parameter), \( \text{dim} = p \times T \).

C_{t}
and \( \text{CSt} \) the posterior scale matrix \( C_{t} = C_{t} \times S_{t} \), with \( \text{dim} = p \times p \times T \), where \( S_{t} \) is a point estimate for the observation variance \( \phi^{-1} \).

R_{t}
and \( \text{RSt} \) the prior scale matrix \( R_{t} = R_{t} \times S_{t-1} \), with \( \text{dim} = p \times p \times T \), where \( S_{t-1} \) is a point estimate for the observation variance \( \phi^{-1} \) at the previous time point.

nt
and \( dt \) the vectors of the updated hyperparameters for the precision \( \phi \) with length \( T \).

S
the vector of the point estimate for the observation variance \( \phi^{-1} \) with length \( T \).

ft
the vector of the one-step forecast location parameter with length \( T \).

Qt
the vector of the one-step forecast scale parameter with length \( T \).

ets
the vector of the standardised forecast residuals with length \( T \), defined as \( (Y_{t} - f_{t}) / \sqrt{Q_{t}} \).

lpl
the vector of the Log Predictive Likelihood with length \( T \).

References


Examples

data("utestdata")
Yt = myts[,1]
Ft = t(cbind(1,myts[,2:5]))
m = dlm.ipl(Yt, Ft, 0.7)

---

### dlm.retro

**Calculate the location and scale parameters for the time-varying coefficients given all the observations.** West, M. & Harrison, J., 1997. Bayesian Forecasting and Dynamic Models. Springer New York.

**Description**


**Usage**

dlm.retro(mt, CSt, RSt, nt, dt)
Arguments

- \( m_t \): the vector or matrix of the posterior mean (location parameter), \( \text{dim} = p \times T \), where \( p \) is the number of thetas (at any time \( t \)) and \( T \) is the number of time points.
- \( \text{CST} \): the posterior scale matrix with \( \text{dim} = p \times p \times T \) (unscaled by the observation variance).
- \( \text{RST} \): the prior scale matrix with \( \text{dim} = p \times p \times T \) (unscaled by the observation variance).
- \( n_t \): vector of the updated hyperparameters for the precision \( \phi \) with length \( T \).
- \( d_t \): vector of the updated hyperparameters for the precision \( \phi \) with length \( T \).

Value

- \( \text{smt} \): the location parameter of the retrospective distribution with dimension \( p \times T \).
- \( \text{sCt} \): the scale matrix of the retrospective distribution with dimension \( p \times p \times T \).

---

**dlmLplCpp**

C++ implementation of the dlm.lpl

---

Description

C++ implementation of the dlm.lpl

Usage

\[
dlmLplCpp(Yt_, Ft_, \delta, m0_, CS0_, n0, d0)
\]

Arguments

- \( Yt_ \): the vector of observed time series.
- \( Ft_ \): the matrix of covariates.
- \( \delta \): discount factor.
- \( m0_ \): the value of the prior mean.
- \( CS0_ \): controls the scaling of the prior variance.
- \( n0 \): prior hyperparameter.
- \( d0 \): prior hyperparameter.
exhaustive.search

**Description**

A function for an exhaustive search, calculates the optimum value of the discount factor.

**Usage**

```r
exhaustive.search(
  Data,
  node,
  nbf = 15,
  delta = seq(0.5, 1, 0.01),
  cpp = TRUE,
  priors = priors.spec()
)
```

**Arguments**

- **Data**
  Dataset with dimension number of time points T x Number of nodes Nn.
- **node**
  The node to find parents for.
- **nbf**
  Log Predictive Likelihood will sum from (and including) this time point.
- **delta**
  a vector of potential values for the discount factor.
- **cpp**
  boolean true (default): fast C++ implementation, false: native R code.
- **priors**
  list with prior hyperparameters.

**Value**

- `model.store` a matrix with the model, LPL and chosen discount factor for all possible models. 
- `runtime` an estimate of the run time of the function, using proc.time().

**Examples**

```r
data("utestdata")
result=exhaustive.search(myts,3)
```


getAdjacency

Get adjacency and associated likelihoods (LPL) and disount factros (df) of winning models.

**Description**

Get adjacency and associated likelihoods (LPL) and disount factros (df) of winning models.

**Usage**

getAdjacency(winner, nodes)

**Arguments**

- **winner**: 2D matrix.
- **nodes**: number of nodes.

**Value**

- adj: 2D adjacency matrix.

getIncompleteNodes

Checks results and returns job number for incomplete nodes.

**Description**

Checks results and returns job number for incomplete nodes.

**Usage**

getIncompleteNodes(path, ids, Nr, Nn)

**Arguments**

- **path**: path to results.
- **ids**: subjects ids.
- **Nr**: Number of runs.
- **Nn**: Number of nodes.

**Value**

- jobs job numbers
getModel

Extract specific parent model with associated df and ME from complete model space.

Description

Extract specific parent model with associated df and ME from complete model space.

Usage

getModel(models, parents)

Arguments

- **models**: a 2D model matrix.
- **parents**: a vector with parent nodes.

Value

mod specific parent model.

Examples

data("utestdata")
r=exhaustive.search(myts,3)
# get model with parents 1, 2, and 4.
m=getModel(r$model.store,c(1,2,4))

getModelNr

Get model number from a set of parents.

Description

Get model number from a set of parents.

Usage

getModelNr(models, parents)

Arguments

- **models**: a 2D model matrix.
- **parents**: a vector with parent nodes.

Value

nr model number.
getWinner

*Get winner network by maximizing log predictive likelihood (LPL) from a set of models.*

Description

Get winner network by maximizing log predictive likelihood (LPL) from a set of models.

Usage

`getWinner(models, nodes)`

Arguments

- **models**: 2D matrix, or 3D models x node.
- **nodes**: number of nodes.

Value

winner array with highest scored model(s).

gplotMat

*Plots network as adjacency matrix.*

Description

Plots network as adjacency matrix.

Usage

```
gplotMat(
  adj,
  title = NULL,
  colMapLabel = NULL,
  hasColMap = NULL,
  lim = c(0, 1),
  gradient = c("white", "orange", "red"),
  nodeLabels = waiver(),
  axisTextSize = 12,
  xAngle = 0,
  titleTextSize = 12,
  barWidth = 1,
  textSize = 12
)
```
mergeModels

Arguments

adj  2D adjacency matrix.
title title.
colMapLabel label for colormap.
hasColMap FALSE turns off color map, default is NULL (on).
lim vector with min and max value, data outside this range will be removed.
gradients gradient colors.
nodeLabels node labels.
axisTextSize text size of the y and x tick labels.
xAngle orientation of the x tick labels.
titleTextSize text size of the title.
barWidth width of the colorbar.
textSize width of the colorbar.

Examples

# Generate some sample binary 5-node network structures for N=20, then compute
# proportion at each edge
N=20
x = array(rbinom(n=5*5*N, size=1, prob=0.30), dim=c(5,5,N))
A = apply(x, c(1,2), mean)

gplotMat(A, title = "network", colMapLabel = '%', barWidth = 0.3)

mergeModels Merges forward and backward model store.

Description

Merges forward and backward model store.

Usage

mergeModels(fw, bw)

Arguments

fw forward model.
bw backward model.

Value

m model store.
**model.generator**

A function to generate all the possible models.

**Description**

A function to generate all the possible models.

**Usage**

`model.generator(Nn, node)`

**Arguments**

- **Nn**
  - number of nodes; the number of columns of the dataset can be used.
- **node**
  - The node to find parents for.

**Value**

output.model = a matrix with dimensions (Nn-1) x number of models, where number of models = $2^{(Nn-1)}$.

**Examples**

```r
m=model.generator(5,1)
```

**myts**

Network simulation data.

**Description**

Simulation 22 5 node net from Smith et al. 2011 (only first subject).
node

Runs exhaustive search on a single node and saves results in txt file.

Description

Runs exhaustive search on a single node and saves results in txt file.

Usage

node(
  X,
  n,
  id = NULL,
  nbf = 15,
  delta = seq(0.5, 1, 0.01),
  cpp = TRUE,
  priors = priors.spec(),
  path = getwd(),
  method = "exhaustive"
)

Arguments

X array with dimensions timeseries x nodes.

n node number.

id subject ID. If set, results are saved to a txt file.

nbf Log Predictive Likelihood will sum from (and including) this time point.

delta a vector of potential values for the discount factor.

cpp boolean true (default): fast C++ implementation, false: native R code.

priors list with prior hyperparameters.

path a path where results are written.

method can be exhaustive (default), forward, backward, or both.

Value

store list with results.
Description

Patel.

Usage

patel(X, lower = 0.1, upper = 0.9, bin = 0.75, TK = 0, TT = 0)

Arguments

X         time x node 2D matrix.
lower     percentile cutoff.
upper     percentile cutoff for 0-1 scaling.
bin       threshold for conversion to binary values.
TK        significance threshold for connection strength kappa.
TT        significance threshold for direction tau.

Value

PT list with strengths kappa, direction tau, and net structure.

Examples

# Generate some sample data
x=array(rnorm(200*5), dim=c(200,5))
p=patel(x)

patel.group

A group is a list containing restructured data from subjects for easier group analysis.

Description

A group is a list containing restructured data from subjects for easier group analysis.

Usage

patel.group(subj)

Arguments

subj  a list of subjects.
Value

   group a list.

Examples

   # create some sample data with 200 samples,  
   # 3 nodes, and 2 subjects
   ts = array(rnorm(200*3*2), dim=c(200,3,2))  
   mysubs=list()  
   mysubs[[1]]=patel(ts[,1])  
   mysubs[[2]]=patel(ts[,2])  
   g=patel.group(mysubs)

perf                       Performance of estimates, such as sensitivity, specificity, and more.

Description

   Performance of estimates, such as sensitivity, specificity, and more.

Usage

   perf(x, true)

Arguments

   x                   estimated binary network matrix.
   true,               true binary network matrix.

Value

   p list with results.

Examples

   trueNet=matrix(c(0,0,0,1,0,0,1,0),3,3)  
   am=matrix(c(0,0,1,0,1,0,1,0),3,3)  
   p=perf(am, trueNet)
**priors.spec**  
*Specify the priors. Without inputs, defaults will be used.*

**Description**

Specify the priors. Without inputs, defaults will be used.

**Usage**

```
priors.spec(m0 = 0, CS0 = 3, n0 = 0.001, d0 = 0.001)
```

**Arguments**

- **m0**  
  the value of the prior mean at time \( t=0 \), scalar (assumed to be the same for all nodes). The default is zero.

- **CS0**  
  controls the scaling of the prior variance matrix \( C_0 \) at time \( t=0 \). The default is 3, giving a non-informative prior for \( C_0 = 3 \times (p \times p) \) identity matrix. \( p \) is the number of thetas.

- **n0**  
  prior hyperparameter of precision \( \phi \sim G(n_0/2; d_0/2) \). The default is a non-informative prior, with \( n_0 = d_0 = 0.001 \). \( n_0 \) has to be higher than 0.

- **d0**  
  prior hyperparameter of precision \( \phi \sim G(n_0/2; d_0/2) \). The default is a non-informative prior, with \( n_0 = d_0 = 0.001 \).

**Details**

At time \( t=0 \), \( \theta_0 | D_0, \phi \sim N(m_0, C_0 \times \phi^{-1}) \), where \( D_0 \) denotes the set of initial information.

**Value**

`priors` a list with the prior hyperparameters. Relevant to `dlm.lpl`, `exhaustive.search`, `node`, `subject`.

**References**


**Examples**

```
pr=priors.spec()
pr=priors.spec(n0=0.002)
```
**prop.nettest**

*Comparing two population proportions on the network with FDR correction.*

**Description**

Comparing two population proportions on the network with FDR correction.

**Usage**

```
prop.nettest(x1, n1, x2, n2, alpha = 0.05, fdr = 0.05)
```

**Arguments**

- `x1`: network matrix with successes in group 1.
- `n1`: sample size group 1.
- `x2`: network matrix with successes in group 2.
- `n2`: sample size group 2.
- `alpha`: alpha level for uncorrected test.
- `fdr`: alpha level for FDR.

**Value**

store List with test statistics and p-values.

---

**pruning**

*Get pruned adjacency network.*

**Description**

Get pruned adjacency network.

**Usage**

```
pruning(adj, models, winner, e = 20)
```

**Arguments**

- `adj`: list with network adjacency from getAdjacency().
- `models`: list of models.
- `winner`: matrix 2D with winning models.
- `e`: bayes factor for network pruning.
Value

thr list with pruned network adjacency.

Examples

data("utestdata")
# select only 3-nodes to speed-up this example
sub=subject(myts[,1:3])
p=pruning(sub$adj, sub$models, sub$winner)

rand.test

Randomization test for Patel's kappa. Creates a distribution of values kappa under the null hypothesis.

Description

Randomization test for Patel's kappa. Creates a distribution of values kappa under the null hypothesis.

Usage

rand.test(X, alpha = 0.05, K = 1000)

Arguments

X        time x node x subjects 3D matrix.
alpha    sign. level
K        number of randomizations, default is 1000.

Value

stat lower and upper significance thresholds.

Examples

# create some sample data with 200 samples,
# 3 nodes, and 2 subjects
xs = array(rnorm(200*3*5), dim=c(200,3,5))
mysubs=list()
mysubs[[1]]=patel(xs[,1])
mysubs[[2]]=patel(xs[,2])
mysubs[[3]]=patel(xs[,3])
mysubs[[4]]=patel(xs[,4])
mysubs[[5]]=patel(xs[,5])
g=patel.group(mysubs)
r=rand.test(rmdiag(g$kappa), K=100)
**read.subject**

*Reads single subject’s network from txt files.*

**Description**

Reads single subject’s network from txt files.

**Usage**

```r
read.subject(path, id, nodes, modelStore = TRUE)
```

**Arguments**

- `path` path.
- `id` identifier to select all subjects’ nodes, e.g. pattern containing subject ID and session number.
- `nodes` number of nodes.
- `modelStore` can be set to false to save memory.

**Value**

store list with results.

---

**reshapeTs**

*Reshapes a 2D concatenated time series into 3D according to no. of subjects and volumes.*

**Description**

Reshapes a 2D concatenated time series into 3D according to no. of subjects and volumes.

**Usage**

```r
reshapeTs(ts, N, V)
```

**Arguments**

- `ts` a 2D time series volumes x nodes.
- `N` No. of subjects.
- `V` No. of volumes.

**Value**

M 3D matrix, time series x nodes x subjects.
Examples

# Let's say subjects are concatenated in a 2D matrix
# (samples x nodes), with each having 200 samples.
# generate some sample data
N=20
Nn=5
x = array(rnorm(200*N*Nn), dim=c(200*N,Nn))
ts = reshapeTs(x,N,200)

rmdiag  \hspace{1cm} \text{Removes diagonal of NA's from matrix.}

Description

Removes diagonal of NA's from matrix.

Usage

rmdiag(M)

Arguments

M \hspace{1cm} \text{Matrix}

Value

matrix with diagonal of 0's.

Examples

M=array(rnorm(3*3), dim=c(3,3))
M[as.logical(diag(3))] = NA
M=rmna(M)

rmna  \hspace{1cm} \text{Removes NAs from matrix.}

Description

Removes NAs from matrix.

Usage

rmna(M)
**Arguments**

M Matrix

**Value**

matrix with NAs removed.

**Examples**

```r
M = array(NA, dim = c(3, 3))
M[1, 2] = 0.9
M = rmna(M)
```

**Description**

Removes reciprocal connections in the lower diagonal of the network matrix.

**Usage**

```r
rmRecipLow(M)
```

**Arguments**

M adjacency matrix

**Value**

M adjacency matrix without reciprocal connections.

---

**scaleTs**

Scaling data. Zero centers and scales the nodes (SD=1).

**Description**

Scaling data. Zero centers and scales the nodes (SD=1).

**Usage**

```r
scaleTs(X)
```

**Arguments**

X time x node 2D matrix, or 3D with subjects as the 3rd dimension.
Stepwise backward non-exhaustive greedy search, calculates the optimum value of the discount factor.

**Usage**

```r
stepwise.backward(
  Data,
  node,
  nbf = 15,
  delta = seq(0.5, 1, 0.01),
  max.break = TRUE,
  priors = priors.spec()
)
```

**Arguments**

- **Data**: Dataset with dimension number of time points T x number of nodes Nn.
- **node**: The node to find parents for.
- **nbf**: The Log Predictive Likelihood will sum from (and including) this time point.
- **delta**: A vector of values for the discount factor.
- **max.break**: If TRUE, the code will break if adding / removing parents does not improve the LPL. If FALSE, the code will continue to the zero parent / all parent model. Default is TRUE.
- **priors**: List with prior hyperparameters.

**Value**

- **model.store**: The parents, LPL and chosen discount factor for the subset of models scored using this method.
**Description**

Stepwise combine

**Usage**

```r
stepwise.combine(
  Data,
  node,
  nbf = 15,
  delta = seq(0.5, 1, 0.01),
  max.break = TRUE,
  priors = priors.spec()
)
```

**Arguments**

- **Data**: Dataset with dimension number of time points T x number of nodes Nn.
- **node**: The node to find parents for.
- **nbf**: The Log Predictive Likelihood will sum from (and including) this time point.
- **delta**: A vector of values for the discount factor.
- **max.break**: If TRUE, the code will break if adding / removing parents does not improve the LPL. If FALSE, the code will continue to the zero parent / all parent model. Default is TRUE.
- **priors**: List with prior hyperparameters.

**Value**

`model.store` The parents, LPL and chosen discount factor for the subset of models scored using this method.

**Description**

Stepwise forward non-exhaustive greedy search, calculates the optimum value of the discount factor.
subject

Usage

```r
stepwise.forward(
  Data,
  node,
  nbf = 15,
  delta = seq(0.5, 1, 0.01),
  max.break = TRUE,
  priors = priors.spec()
)
```

Arguments

- **Data**
  - Dataset with dimension number of time points T x number of nodes Nn.

- **node**
  - The node to find parents for.

- **nbf**
  - The Log Predictive Likelihood will sum from (and including) this time point.

- **delta**
  - A vector of values for the discount factor.

- **max.break**
  - If TRUE, the code will break if adding / removing parents does not improve the LPL. If FALSE, the code will continue to the zero parent / all parent model. Default is TRUE.

- **priors**
  - List with prior hyperparameters.

Value

- **model.store**
  - The parents, LPL and chosen discount factor for the subset of models scored using this method.

Description

Estimate subject’s full network: runs exhaustive search on very node.

Usage

```r
subject(
  X,
  id = NULL,
  nbf = 15,
  delta = seq(0.5, 1, 0.01),
  cpp = TRUE,
  priors = priors.spec(),
  path = getwd(),
  method = "exhaustive"
)
```
**symmetric**

**Arguments**

- **X** array with dimensions timeseries x nodes.
- **id** subject ID. If set, results are saved to a txt file.
- **nbf** Log Predictive Likelihood will sum from (and including) this time point.
- **delta** a vector of potential values for the discount factor.
- **cpp** boolean true (default): fast C++ implementation, false: native R code.
- **priors** list with prior hyperparameters.
- **path** a path where results are written.
- **method** either exhaustive, forward, backward, or both.

**Value**

store list with results.

**Examples**

```r
data("utestdata")
# select only 3-nodes to speed-up this example
sub=subject(myts[,1:3])
sub=subject(myts[,1:3], method="both")
```

---

**symmetric** turns asymmetric network into a symmetric network. Helper function to determine the detection of a connection while ignoring directionality.

**Description**

Turns asymmetric network into a symmetric network. Helper function to determine the detection of a connection while ignoring directionality.

**Usage**

`symmetric(M)`

**Arguments**

- **M** 3D matrix nodes x nodes x subjects

**Value**

3D matrix nodes x nodes x subjects
Examples

```r
M = array(NA, dim = c(3, 3, 2))
M[, , 1] = matrix(c(0, 0, 0, 1, 0, 0, 0, 1, 0), 3, 3)
M[, , 2] = matrix(c(0, 0, 0, 1, 0, 0, 0, 0, 0), 3, 3)
M_ = symmetric(M)
```

---

`ttest.nettest`  
*Comparing connectivity strength of two groups with FDR correction.*

---

Description

Comparing connectivity strength of two groups with FDR correction.

Usage

`ttest.nettest(m, g, alpha = 0.05, fdr = 0.05, perm = FALSE, n_perm = 9999)`

Arguments

- `m`: matrix with `Nn x Nn x N`.
- `g`: group assignment, vector of type factor of size `N`.
- `alpha`: alpha level for uncorrected test.
- `fdr`: FDR alpha level.
- `perm`: optional permutation test, default is false.
- `n_perm`: number of permutations.

Value

`store` List with test statistics and p-values.

---

`utestdata`  
*Results from v.1.0 for unit tests.*

---

Description

Some LPL values (n2 parent of n1 Simulation 22) to test against.
Index

binom.nettest, 2
center, 3
cor2adj, 4
corTs, 4
dgm.group, 5
diag.delta, 6
dlm.lpl, 6, 18
dlm.retro, 7
dlmLplCpp, 8
exhaustive.search, 9, 18
getAdjacency, 10
getIncompleteNodes, 10
getModel, 11
getModelNr, 11
ggetWinner, 12
gplotMat, 12
mergeModels, 13
model.generator, 14
myts, 14
node, 15, 18
patel, 16
patel.group, 16
perf, 17
priors.spec, 18
prop.nettest, 19
pruning, 19
rand.test, 20
read.subject, 21
reshapeTs, 21
rmdiag, 22
rmna, 22
rmRecipLow, 23
scaleTs, 23
stepwise.backward, 24
stepwise.combine, 25
stepwise.forward, 25
subject, 18, 26
symmetric, 27
ttest.nettest, 28
utestdata, 28