Package: DEGraph (via r-universe)

November 24, 2026

Title Two-sample tests on a graph

Version 1.66.0

Date 2012-04-27

Author Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

Maintainer Laurent Jacob <laurent.jacob@gmail.com>

Description DEGraph implements recent hypothesis testing methods which directly assess whether a particular gene network is differentially expressed between two conditions. This is to be contrasted with the more classical two-step approaches which first test individual genes, then test gene sets for enrichment in differentially expressed genes. These recent methods take into account the topology of the network to yield more powerful detection procedures. DEGraph provides methods to easily test all KEGG pathways for differential expression on any gene expression data set and tools to visualize the results.

License GPL-3

LazyLoad yes

Imports graph, KEGGgraph, lattice, mvtnorm, R.methodsS3, RBGL, Rgraphviz, rrcov, NCIgraph

Suggests corpcor, fields, graph, KEGGgraph, lattice, marray, RBGL, rrcov, Rgraphviz, NCIgraph

Depends R (>= 2.10.0), R.utils

biocViews Microarray, DifferentialExpression, GraphAndNetwork, Network, NetworkEnrichment, DecisionTree

Config/Bioconductor/UnsupportedPlatforms windows, macosx

Config/pak/sysreqs
cmake make libicu-dev libuv1-dev libxml2-dev libssl-dev python3 libzmq3-dev zlib1g-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:30:34 UTC

RemoteUrl https://github.com/bioc/DEGraph

RemoteRef RELEASE_3_24

RemoteSha 415596b268ce1e1a19bcd4b8c4a60f59167b6300

Contents

AN.test . . . 2
annLoi2008 . . . 4
BS.test . . . 5
classLoi2008 . . . 7
exprLoi2008 . . . 8
getConnectedComponentList . . . 9
getKEGGPathways . . . 10
getSignedGraph . . . 11
graph.T2.test . . . 12
grListKEGG . . . 14
hyper.test . . . 14
laplacianFromA . . . 16
plotValuedGraph . . . 18
randomWAMGraph . . . 20
testOneConnectedComponent . . . 21
testOneGraph . . . 23
twoSampleFromGraph . . . 25
writeAdjacencyMatrix2KGML . . . 26

Index **29**

`AN.test` *Performs the Adaptive Neyman test of Fan and Lin (1998)*

Description

Performs the Adaptive Neyman test of Fan and Lin (1998).

Usage

```
AN.test(X1, X2, candK=1:ncol(X1), na.rm=FALSE)
```

Arguments

`X1`	A n1 x p `matrix`, observed data for class 1: p variables, n1 observations.
`X2`	A n2 x p `matrix`, observed data for class 2: p variables, n2 observations.
`candK`	A `vector`, candidate values for the true number of Fourier components.
`na.rm`	A `logical` value indicating whether variables with `NA` in at least one of the n1 + n2 observations should be discarder before the test is performed.

Value

A `list` with class "htest" containing the following components:

statistic A `numeric` value, the test statistic.

p.value A `numeric` value, the corresponding p-value.

kstar A `numeric` value, the estimated true number of Fourier components.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

See Also

BS.test() graph.T2.test() hyper.test()

Examples

```
library("KEGGgraph")
## library("NCIgraph")
library("rrcov")

data("Loi2008_DEGraphVignette")
exprData <- exprLoi2008
classData <- classLoi2008
rn <- rownames(exprData)

## Retrieve expression levels data for genes from one KEGG pathway
gr <- grListKEGG[[1]]
gids <- translateKEGGID2GeneID(nodes(gr))
mm <- match(gids, rownames(exprData))

## Keep genes from the graph that are present in the expression data set
idxs <- which(!is.na(mm))
gr <- subGraph(nodes(gr)[idxs], gr)

idxs <- which(is.na(mm))
if(length(idxs)) {
  print("Gene ID not found in expression data: ")
  str(gids[idxs])
}
dat <- exprData[na.omit(mm), ]
str(dat)

X1 <- t(dat[, classData==0])
X2 <- t(dat[, classData==1])

## DEGraph T2 test
res <- testOneGraph(gr, exprData, classData, verbose=TRUE, prop=0.2)

## T2 test (Hotelling)
rT2 <- T2.test(X1, X2)
str(rT2)

## Adaptive Neyman test
rAN <- AN.test(X1, X2, na.rm=TRUE)
str(rAN)

## Adaptive Neyman test from Fan and Lin (1998)
rAN <- AN.test(X1, X2, na.rm=TRUE)
str(rAN)
```

```
## Test from Bai and Saranadasa (1996)
rBS <- BS.test(X1, X2, na.rm=TRUE)
str(rBS)

## Hypergeometric test
pValues <- apply(exprData, 1, FUN=function(x) {
  tt <- t.test(x[classData==0], x[classData==1])
  tt$p.value
})
str(pValues)
names(pValues) <- rownames(exprData)
rHyper <- hyper.test(pValues, gids, thr=0.01)
str(rHyper)
```

---

annLoi2008 *Annotation data used in the DEGraph package vignette*

Description

This data set gives NCBI, Hugo and alternative gene symbols along with the cytoband and description for the 227 genes used in the DEGraph package vignette. This comes from the 15737 gene, 255 patient dataset of Loi et al. (2008) which was used to study resistance to tamoxifen treatment in hormone-dependent breast cancer.

Usage

```
annLoi2008
```

Format

A matrix of 227 lines and 5 columns.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

Source

Loi et al., *Predicting prognosis using molecular profiling in estrogen receptor-positive breast cancer treated with tamoxifen.* BMC Genomics, 9(1):239, 2008.

References

Loi et al., *Predicting prognosis using molecular profiling in estrogen receptor-positive breast cancer treated with tamoxifen.* BMC Genomics, 9(1):239, 2008.

Examples

```
data("Loi2008_DEGraphVignette")

dim(annLoi2008)
head(annLoi2008)
```

BS.test *Performs the test of Bai and Saranadasa (1996)*

Description

Performs the test of Bai and Saranadasa (1996).

Usage

```
BS.test(X1, X2, na.rm=FALSE)
```

Arguments

X1	A n1 x p matrix, observed data for class 1: p variables, n1 observations.
X2	A n2 x p matrix, observed data for class 2: p variables, n2 observations.
na.rm	A logical value indicating whether variables with NA in at least one of the n1 + n2 observations should be discarder before the test is performed.

Value

A list with class "htest" containing the following components:

statistic A numeric value, the test statistic.

p.value A numeric value, the corresponding p-value.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

See Also

AN.test() graph.T2.test() hyper.test()

Examples

```
library("KEGGgraph")
## library("NCIgraph")
library("rrcov")

data("Loi2008_DEGraphVignette")
exprData <- exprLoi2008
classData <- classLoi2008
rn <- rownames(exprData)

## Retrieve expression levels data for genes from one KEGG pathway
gr <- grListKEGG[[1]]
gids <- translateKEGGID2GeneID(nodes(gr))
mm <- match(gids, rownames(exprData))

## Keep genes from the graph that are present in the expression data set
idxs <- which(!is.na(mm))
gr <- subGraph(nodes(gr)[idxs], gr)

idxs <- which(is.na(mm))
if(length(idxs)) {
  print("Gene ID not found in expression data: ")
  str(gids[idxs])
}
dat <- exprData[na.omit(mm), ]
str(dat)

X1 <- t(dat[, classData==0])
X2 <- t(dat[, classData==1])

## DEGraph T2 test
res <- testOneGraph(gr, exprData, classData, verbose=TRUE, prop=0.2)

## T2 test (Hotelling)
rT2 <- T2.test(X1, X2)
str(rT2)

## Adaptive Neyman test
rAN <- AN.test(X1, X2, na.rm=TRUE)
str(rAN)

## Adaptive Neyman test from Fan and Lin (1998)
rAN <- AN.test(X1, X2, na.rm=TRUE)
str(rAN)

## Test from Bai and Saranadasa (1996)
rBS <- BS.test(X1, X2, na.rm=TRUE)
str(rBS)

## Hypergeometric test
pValues <- apply(exprData, 1, FUN=function(x) {
  tt <- t.test(x[classData==0], x[classData==1])
```

```
  tt$p.value
})
str(pValues)
names(pValues) <- rownames(exprData)
rHyper <- hyper.test(pValues, gids, thr=0.01)
str(rHyper)
```

`classLoi2008` *Tamoxifen treatment resistance status data used in the DEGraph package vignette*

Description

This data set gives resistance status data for the 255 patients used in the DEGraph package vignette. This comes from the 15737 gene, 255 patient dataset of Loi et al. (2008) which was used to study resistance to tamoxifen treatment in hormone-dependent breast cancer.

Usage

```
classLoi2008
```

Format

A vector of 255 elements which are either 0 (resistance to treatment) or 1 (sensitivity to treatment).

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

Source

Loi et al., *Predicting prognosis using molecular profiling in estrogen receptor-positive breast cancer treated with tamoxifen.* BMC Genomics, 9(1):239, 2008.

References

Loi et al., *Predicting prognosis using molecular profiling in estrogen receptor-positive breast cancer treated with tamoxifen.* BMC Genomics, 9(1):239, 2008.

Examples

```
data("Loi2008_DEGraphVignette")

dim(classLoi2008)
head(classLoi2008)
```

exprLoi2008 *Gene expression data used in the DEGraph package vignette*

Description

This data set gives gene expression data for a subset of 227 genes used in the DEGraph package vignette. This comes from the 15737 gene, 255 patient dataset of Loi et al. (2008) which was used to study resistance to tamoxifen treatment in hormone-dependent breast cancer.

Usage

```
exprLoi2008
```

Format

A matrix of 227 lines and 255 columns.

Details

The original data set corresponds to data processed by RMA and median-centered as available from the GSE6532 GEO archive: http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE6532.

These data were summarized from the probe set level to the gene level as follows. The expression level of a gene was defined as the expression level of the probe set with largest alignment score among all probe sets mapping to this gene according to the annotation in GSE6532. When the largest alignment score was achieved by several probe sets, the median expression level of those probe sets was taken.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

Source

Loi et al., *Predicting prognosis using molecular profiling in estrogen receptor-positive breast cancer treated with tamoxifen.* BMC Genomics, 9(1):239, 2008.

References

Loi et al., *Predicting prognosis using molecular profiling in estrogen receptor-positive breast cancer treated with tamoxifen.* BMC Genomics, 9(1):239, 2008.

Examples

```
data("Loi2008_DEGraphVignette")

dim(exprLoi2008)
head(exprLoi2008)
```

getConnectedComponentList *Given a graph, returns a list of its connected components (which are also graph objects), ordered by decreasing number of nodes*

Description

Given a graph, returns a list of its connected components (which are also graph objects), ordered by decreasing number of nodes.

Usage

```
getConnectedComponentList(graph, verbose=FALSE)
```

Arguments

graph	A graph object.
verbose	If TRUE, extra information is output.

Value

A list containing a graph object for each connected component of the input graph, ordered by decreasing number of nodes

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

See Also

connectedComp.

Examples

```
data("Loi2008_DEGraphVignette")
exprData <- exprLoi2008
rn <- rownames(exprData)

## Retrieve expression levels data for genes from one KEGG pathway
graph <- grListKEGG[[1]]
pname <- attr(graph, "label")
cat(verbose, "Pathway name: ", pname)

sgraph <- getSignedGraph(graph, verbose=TRUE)
print(sgraph)

graphList <- getConnectedComponentList(graph, verbose=TRUE)
print(graphList)
```

getKEGGPathways *Builds a graph for each of the KEGG pathways*

Description

Builds a graph for each of the KEGG pathways.

Usage

```
getKEGGPathways(path=NULL, rootPath="networkData/ftp.genome.jp/pub/kegg/xml/kgml", organism="hsa", m
```

Arguments

path	A character value, the local _full_ path of KGML data.
rootPath	A character value, the local _root_ path of KGML data.
organism	A character value specifying the organism whose pathways should be considered. Defaults to "hsa" (Homo Sapiens).
metaTag	A character value, specifying the type of pathways to be considered ("metabolic" or "non-metabolic"). Defaults to "non-metabolic".
pattern	An optional character value specifying a file name pattern to look for.
verbose	If TRUE, extra information is output.

Details

If 'path' is supplied, KGML files in this directory are loaded. Otherwise, KGML files are assumed to be in <rootPath>/<metaTag>/"organisms"/<organism>, which mirrors the structure of the KEGG KGML file repository.

Value

A list containing a graph object for each KEGG pathway with at least one edge.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

See Also

parseKGML KEGGpathway2Graph

Examples

```
library("Rgraphviz")
library("KEGGgraph")

## example of KGML files
path <- system.file("extdata", package="KEGGgraph")
grList <- getKEGGPathways(path=path, verbose=TRUE)
print(grList)

graph <- grList[[1]]
plotKEGGgraph(graph)

## Not run:
  ## Download all human KGML pathways locally
  pathname <- system.file("downloadScripts", "downloadKeggXmlFiles.R", package="DEGraph")
  source(pathname)

  ## Load some of them
  grList <- getKEGGPathways(pattern="040", verbose=TRUE)
  print(grList)

  graph <- grList[[1]]
  plotKEGGgraph(graph)

## End(Not run)
```

`getSignedGraph` *Given a graph, builds a signed version of the adjacency matrix taking into account the type of interaction (e.g., activation or inhibition)*

Description

Given a graph, builds a signed version of the adjacency matrix taking into account the type of interaction (e.g., activation or inhibition).

Usage

```
getSignedGraph(graph, positiveInteractionLabels=c("activation", "expression"), negativeInteractionLa
```

Arguments

`graph`	A `graph` object.
`positiveInteractionLabels`	A `character vector` specifying which interaction labels correspond to positive interactions. Defaults to 'c("activation", "expression")'.
`negativeInteractionLabels`	A `character vector` specifying which interaction labels correspond to negative interactions. Defaults to 'c("inhibition", "repression")'.
`verbose`	If `TRUE`, extra information is output.

Value

This function returns a squared matrix whose (i,j) entry is:

0 if edges i and j are not connected

1 if edges i and j are connected by a positive interaction

-1 if edges i and j are connected by a negative interaction.

By construction, the absolute value of this matrix is the adjacency matrix of the graph. Edges which cannot interpreted as corresponding to a positive or a negative interaction are marked as not connected.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

Examples

```
data("Loi2008_DEGraphVignette")
exprData <- exprLoi2008
rn <- rownames(exprData)

## Retrieve expression levels data for genes from one KEGG pathway
graph <- grListKEGG[[1]]
pname <- attr(graph, "label")
cat(verbose, "Pathway name: ", pname)

sgraph <- getSignedGraph(graph, verbose=TRUE)
print(sgraph)

graphList <- getConnectedComponentList(graph, verbose=TRUE)
print(graphList)
```

---

graph.T2.test *Performs the Hotelling T2 test in Fourier space*

Description

Performs the Hotelling T2 test in Fourier space.

Usage

```
graph.T2.test(X1, X2, G=NULL, lfA=NULL, ..., k=ncol(X1))
```

Arguments

X1	A n1 x p numeric matrix, observed data for class 1: p variables, n1 observations.
X2	A n2 x p numeric matrix, observed data for class 2: p variables, n2 observations.
G	An object of class graphAM or graphNEL, the graph to be used in the two-sample test.
lfA	A list returned by laplacianFromA(), containing the Laplacian eigen vectors and eigen values
...	Further arguments to be passed to laplacianFromA().
k	A numeric value, number of Fourier components retained for the test.

Value

A list with class "htest", as returned by T2.test.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

See Also

T2.test graphAM

Examples

```
library("rrcov")

## Some parameters
n1 <- n2 <- 20
nnodes <- nedges <- 20
k <- 3
ncp <- 0.5
sigma <- diag(nnodes)/sqrt(nnodes)


## Build graph, decompose laplacian
G <- randomWAMGraph(nnodes=nnodes,nedges=nedges)
A <- G@adjMat
lfA <- laplacianFromA(A,ltype="unnormalized")
U <- lfA$U
l <- lfA$l

## Build two samples with smooth mean shift
X <- twoSampleFromGraph(n1,n2,shiftM2=ncp,sigma,U=U,k=k)

## Do hypothesis testing
t <- T2.test(X$X1,X$X2) # Raw T-square
print(t$p.value)
```

```
tu <- graph.T2.test(X$X1,X$X2,lfA=lfA,k=k) # Filtered T-squares
print(tu$p.value)
```

---

grListKEGG *KEGG networks used in the DEGraph package vignette*

Description

This data set gives KEGGgraph objects for two KEGG non-metabolic pathways ("Natural killer cell mediated cytotoxicity" and "Insulin signaling pathway").

Usage

```
grListKEGG
```

Format

A list of two elements.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

Examples

```
library("Rgraphviz")

data("Loi2008_DEGraphVignette")

grListKEGG
plot(grListKEGG[[1]])
```

---

hyper.test *Performs an hypergeometric test of enrichment of a set of hypotheses in significant elements*

Description

Performs an hypergeometric test of enrichment of a set of hypotheses in significant elements.

Usage

```
hyper.test(p.values, testSet, thr=0.001, universe=length(p.values), verbose=FALSE)
```

Arguments

p.values	A named numeric vector giving the p-values of all tested elements.
testSet	A character vector giving the ids of the elements in the tested set. Elements of 'testSet' must have a match in 'names(p.values)'.
thr	A numeric value between 0 and 1 giving the threshold on p-values at which an element is declared to be significant.
universe	An integer value giving the number of elelments in the considered universe. Defaults to 'length(p.values)'.
verbose	If TRUE, extra information is output.

Value

A list with class "htest" containing the following components:

statistic A numeric value, the test statistic.

p.value A numeric value, the corresponding p-value.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

See Also

AN.test() BS.test() graph.T2.test()

Examples

```
library("KEGGgraph")
## library("NCIgraph")
library("rrcov")

data("Loi2008_DEGraphVignette")
exprData <- exprLoi2008
classData <- classLoi2008
rn <- rownames(exprData)

## Retrieve expression levels data for genes from one KEGG pathway
gr <- grListKEGG[[1]]
gids <- translateKEGGID2GeneID(nodes(gr))
mm <- match(gids, rownames(exprData))

## Keep genes from the graph that are present in the expression data set
idxs <- which(!is.na(mm))
gr <- subGraph(nodes(gr)[idxs], gr)

idxs <- which(is.na(mm))
if(length(idxs)) {
  print("Gene ID not found in expression data: ")
  str(gids[idxs])
}
```

```
dat <- exprData[na.omit(mm), ]
str(dat)

X1 <- t(dat[, classData==0])
X2 <- t(dat[, classData==1])

## DEGraph T2 test
res <- testOneGraph(gr, exprData, classData, verbose=TRUE, prop=0.2)

## T2 test (Hotelling)
rT2 <- T2.test(X1, X2)
str(rT2)

## Adaptive Neyman test
rAN <- AN.test(X1, X2, na.rm=TRUE)
str(rAN)

## Adaptive Neyman test from Fan and Lin (1998)
rAN <- AN.test(X1, X2, na.rm=TRUE)
str(rAN)

## Test from Bai and Saranadasa (1996)
rBS <- BS.test(X1, X2, na.rm=TRUE)
str(rBS)

## Hypergeometric test
pValues <- apply(exprData, 1, FUN=function(x) {
  tt <- t.test(x[classData==0], x[classData==1])
  tt$p.value
})
str(pValues)
names(pValues) <- rownames(exprData)
rHyper <- hyper.test(pValues, gids, thr=0.01)
str(rHyper)
```

`laplacianFromA` *Calculates the Laplacian associated to an adjacency matrix*

Description

Calculates the Laplacian associated to an adjacency matrix.

Usage

```
laplacianFromA(A, k=1, ltype=c("meanInfluence", "normalized", "unnormalized", "totalInfluence"))
```

Arguments

A	The adjacency matrix of the graph.
k	...
ltype	A character value specifying the type of Laplacian to be calculated. Defaults to meanInfluence.

Value

A list containing the following components:

U Eigenvectors of the graph Laplacian.

l Eigenvalues of the graph Laplacian

kIdx Multiplicity of '0' as eigenvalue.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

Examples

```
library("KEGGgraph")
library("rrcov")

## Create a random graph
graph <- randomWAMGraph(nnodes=5, nedges=7, verbose=TRUE)
plot(graph)

## Retrieve its adjacency matrix
A <- graph@adjMat

## write it to KGML file
grPathname <- "randomWAMGraph.xml"
writeAdjacencyMatrix2KGML(A, pathname=grPathname, verbose=TRUE, overwrite=TRUE)

## read it from file
gr <- parseKGML2Graph(grPathname)

## Two examples of Laplacians from the same graph
lapMI <- laplacianFromA(A, ltype="meanInfluence")
print(lapMI)

lapN <- laplacianFromA(A, ltype="normalized")
print(lapN)

U <- lapN$U
p <- nrow(A)
sigma <- diag(p)/sqrt(p)

X <- twoSampleFromGraph(100, 120, shiftM2=1, sigma, U=U, k=3)
```

```
## T2
t <- T2.test(X$X1,X$X2)
str(t)

tu <- graph.T2.test(X$X1, X$X2, lfA=lapMI, k=3)
str(tu)
```

---

plotValuedGraph *Plots a graph with nodes colored according to a quantitative variable*

Description

Plots a graph with nodes colored according to a quantitative variable.

Usage

```
plotValuedGraph(graph, values=NULL, nodeLabels=nodes(graph), qMax=0.95, colorPalette=heat.colors(10),
```

Arguments

graph	A graph object.
values	A named vector of numeric values according to which the graph nodes should be colored.
nodeLabels	A character vector of the same length and in the same order as 'nodes(graph)': node labels to be displayed. Defaults to 'nodes(graph)'.
qMax	A numeric value, fraction of the data to be truncated in order to avoid outliers.
colorPalette	A character vector, the set of colors to be used.
adjustColorRange	A logical value. If TRUE, the color range is adjusted to the range of values of nodes actually present in the graph. Defaults to FALSE, i.e. the color range spans range(values) regardless of which nodes are present in the graph.
symmetrizeArrows	A logical value. If TRUE, arrow tails are drawn as the corresponding arrow heads. Defaults to FALSE.
height	A numeric value, the (common) size of nodes.
lwd	A numeric value, the (common) width of edges.
cex	A numeric value, the relative size of the text for gene names.
...	Further arguments to be passed to 'edgeRenderInfo' and 'nodeRenderInfo'.
verbose	If TRUE, extra information is output.

Value

A list containing the following components:

graph The 'graph' object as plotted.

breaks The break points in the supplied values (can be used for plotting a legend).

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

See Also

plotKEGGgraph plot()

Examples

```
library("Rgraphviz")
library("KEGGgraph")
## library("NCIgraph")

data("Loi2008_DEGraphVignette")
exprData <- exprLoi2008
classData <- classLoi2008
annData <- annLoi2008

rn <- rownames(exprData)

## Retrieve expression levels data for genes from one KEGG pathway
graph <- grListKEGG[[1]]
pname <- attr(graph, "label")
print(pname)

## DEGraph T2 test
resList <- testOneGraph(graph, exprData, classData, verbose=TRUE, prop=0.2)

## Largest connected component
res <- resList[[1]]
gr <- res$graph

## individual t statistics
shift <- apply(exprData, 1, FUN=function(x) {
  tt <- t.test(x[classData==0], x[classData==1])
  tt$statistic
})
names(shift) <- translateGeneID2KEGGID(names(shift))

## color palette
if (require(marray)) {
  pal <- maPalette(low="red", high="green", mid="black", k=100)
} else {
  pal <- heat.colors(100)
}

## plot results
dn <- getDisplayName(gr, shortLabel=TRUE)
mm <- match(translateKEGGID2GeneID(nodes(gr)), rownames(annData))
dn <- annData[mm, "NCBI.gene.symbol"]

pvg <- plotValuedGraph(gr, values=shift, nodeLabels=dn, qMax=0.95, colorPalette=pal, height=40, lwd=1, verbose=TR
```

```
title(pname)

txt1 <- sprintf("p(T2)=%s", signif(res$p.value[1], 2))
txt2 <- sprintf("p(T2F[%s])=%s", res$k, signif(res$p.value[2]))
txt <- paste(txt1, txt2, sep="\n")
stext(side=3, pos=1, txt)
if (require(fields)) {
 image.plot(legend.only=TRUE, zlim=range(pvg$breaks), col=pal, legend.shrink=0.3, legend.width=0.8, legend.lab='
}
```

---

randomWAMGraph *Generates a random graph*

Description

Generates a random graph.

Usage

```
randomWAMGraph(nnodes=5, nedges=nnodes, verbose=FALSE)
```

Arguments

nnodes	A numeric value, the desired number of nodes.
nedges	A numeric value, the desired number of edges.
verbose	If TRUE, extra information is output.

Value

An object of class graphAM.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

See Also

graphAM.

Examples

```
library("KEGGgraph")
library("rrcov")

## Create a random graph
graph <- randomWAMGraph(nnodes=5, nedges=7, verbose=TRUE)
plot(graph)

## Retrieve its adjacency matrix
```

```
A <- graph@adjMat

## write it to KGML file
grPathname <- "randomWAMGraph.xml"
writeAdjacencyMatrix2KGML(A, pathname=grPathname, verbose=TRUE, overwrite=TRUE)

## read it from file
gr <- parseKGML2Graph(grPathname)

## Two examples of Laplacians from the same graph
lapMI <- laplacianFromA(A, ltype="meanInfluence")
print(lapMI)

lapN <- laplacianFromA(A, ltype="normalized")
print(lapN)

U <- lapN$U
p <- nrow(A)
sigma <- diag(p)/sqrt(p)

X <- twoSampleFromGraph(100, 120, shiftM2=1, sigma, U=U, k=3)

## T2
t <- T2.test(X$X1,X$X2)
str(t)

tu <- graph.T2.test(X$X1, X$X2, lfA=lapMI, k=3)
str(tu)
```

`testOneConnectedComponent` *Applies a series of two-sample tests to a connected graph using various statistics*

Description

Applies a series of two-sample tests to a connected graph using various statistics.

Usage

```
testOneConnectedComponent(graph, data, classes, ..., prop=0.2, verbose=FALSE)
```

Arguments

`graph`	A `graph` object.
`data`	A '`numeric` `matrix` (size: number 'p' of genes x number 'n' of samples) of gene expression.
`classes`	A `character` `vector` (length: 'n') of class assignments.
`...`	Further arguments to be passed to `laplacianFromA`().

prop	A numeric value, percentage of components retained for Fourier and PCA.
verbose	If TRUE, extra information is output.

Details

This function performs the test, assuming that all genes in the graph are represented in the expression data set, in order not to have to modify the graph topology.

Interaction signs are used if available in the graph ('getSignedGraph' is not called here, in order not to have to modify the graph topology.).

The graph given as input has to have only one connex component. It can be retrieved from the output of getConnectedComponentList().

Value

A structured list containing the p-values of the tests, the graph object of the connected component and the number of retained Fourier dimensions.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

See Also

testOneGraph() getConnectedComponentList()

Examples

```
library("rrcov")

## Some parameters
n1 <- n2 <- 20
nnodes <- nedges <- 20
k <- 3
ncp <- 0.5
sigma <- diag(nnodes)/sqrt(nnodes)


## Build graph, decompose laplacian
G <- randomWAMGraph(nnodes=nnodes,nedges=nedges)
A <- G@adjMat
lfA <- laplacianFromA(A,ltype="unnormalized")
U <- lfA$U
l <- lfA$l

## Build two samples with smooth mean shift
X <- twoSampleFromGraph(n1,n2,shiftM2=ncp,sigma,U=U,k=k)

## Do hypothesis testing
t <- T2.test(X$X1,X$X2) # Raw T-square
print(t$p.value)
tu <- graph.T2.test(X$X1,X$X2,lfA=lfA,k=k) # Filtered T-squares
```

```
print(tu$p.value)
```

`testOneGraph` *Applies a serie of two-sample tests to each connected component of a graph using various statistics*

Description

Applies a serie of two-sample tests to each connected component of a graph using various statistics.

Usage

```
testOneGraph(graph, data, classes, useInteractionSigns=TRUE, ..., verbose=FALSE)
```

Arguments

`graph`	A `graph` object.
`data`	A 'matrix' (size: number 'p' of genes x number 'n' of samples) of gene expression.
`classes`	A 'vector' (length: 'n') of class assignments.
`useInteractionSigns`	A `logical` value indicating whether the sign of interaction should be taken into account.
`...`	Further arguments to be passed to testOneConnectedComponent.
`verbose`	If `TRUE`, extra information is output.

Value

A structured `list` containing the p-values of the tests, the `graph` object of the connected component and the number of retained Fourier dimensions.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

See Also

`testOneConnectedComponent`()

Examples

```
library("Rgraphviz")
library("KEGGgraph")
## library("NCIgraph")

data("Loi2008_DEGraphVignette")
exprData <- exprLoi2008
classData <- classLoi2008
annData <- annLoi2008

rn <- rownames(exprData)

## Retrieve expression levels data for genes from one KEGG pathway
graph <- grListKEGG[[1]]
pname <- attr(graph, "label")
print(pname)

## DEGraph T2 test
resList <- testOneGraph(graph, exprData, classData, verbose=TRUE, prop=0.2)

## Largest connected component
res <- resList[[1]]
gr <- res$graph

## individual t statistics
shift <- apply(exprData, 1, FUN=function(x) {
  tt <- t.test(x[classData==0], x[classData==1])
  tt$statistic
})
names(shift) <- translateGeneID2KEGGID(names(shift))

## color palette
if (require(marray)) {
  pal <- maPalette(low="red", high="green", mid="black", k=100)
} else {
  pal <- heat.colors(100)
}

## plot results
dn <- getDisplayName(gr, shortLabel=TRUE)
mm <- match(translateKEGGID2GeneID(nodes(gr)), rownames(annData))
dn <- annData[mm, "NCBI.gene.symbol"]

pvg <- plotValuedGraph(gr, values=shift, nodeLabels=dn, qMax=0.95, colorPalette=pal, height=40, lwd=1, verbose=TR
title(pname)

txt1 <- sprintf("p(T2)=%s", signif(res$p.value[1], 2))
txt2 <- sprintf("p(T2F[%s])=%s", res$k, signif(res$p.value[2]))
txt <- paste(txt1, txt2, sep="\n")
stext(side=3, pos=1, txt)
if (require(fields)) {
 image.plot(legend.only=TRUE, zlim=range(pvg$breaks), col=pal, legend.shrink=0.3, legend.width=0.8, legend.lab="
```

```
}
```

`twoSampleFromGraph` *Given a basis (typically the eigenvectors of a graph Laplacian), builds two multivariate normal samples with mean shift located in the first elements of the basis*

Description

Given a basis (typically the eigenvectors of a graph Laplacian), builds two multivariate normal samples with mean shift located in the first elements of the basis.

Usage

```
twoSampleFromGraph(n1=20, n2=n1, shiftM2=0, sigma, U, k=ceiling(ncol(U)/3))
```

Arguments

`n1`	An `integer` value specifying the number of points in the first sample.
`n2`	An `integer` value specifying the number of points in the second sample.
`shiftM2`	A `numeric` value giving the desired squared Mahalanobis norm of the mean shift between the two samples.
`sigma`	A matrix giving the covariance structure of each sample.
`U`	A matrix giving the desired basis.
`k`	An `integer` value giving the number of basis elements in which the mean shift must be located.

Value

A `list` with named elements:

X1 The first sample in the original basis (before transformation by U).

X2 The second sample in the original basis (before transformation by U).

X1 The first sample in the specified basis (after transformation by U).

X2 The second sample in the specified basis (after transformation by U).

mu1 The population mean of F1

mu2 The population mean of F2

diff mu1 - mu2

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

Examples

```
library("KEGGgraph")
library("rrcov")

## Create a random graph
graph <- randomWAMGraph(nnodes=5, nedges=7, verbose=TRUE)
plot(graph)

## Retrieve its adjacency matrix
A <- graph@adjMat

## write it to KGML file
grPathname <- "randomWAMGraph.xml"
writeAdjacencyMatrix2KGML(A, pathname=grPathname, verbose=TRUE, overwrite=TRUE)

## read it from file
gr <- parseKGML2Graph(grPathname)

## Two examples of Laplacians from the same graph
lapMI <- laplacianFromA(A, ltype="meanInfluence")
print(lapMI)

lapN <- laplacianFromA(A, ltype="normalized")
print(lapN)

U <- lapN$U
p <- nrow(A)
sigma <- diag(p)/sqrt(p)

X <- twoSampleFromGraph(100, 120, shiftM2=1, sigma, U=U, k=3)

## T2
t <- T2.test(X$X1,X$X2)
str(t)

tu <- graph.T2.test(X$X1, X$X2, lfA=lapMI, k=3)
str(tu)
```

`writeAdjacencyMatrix2KGML`

Writes an adjacency matrix into an XML file

Description

Writes an adjacency matrix into an XML file.

Usage

```
writeAdjacencyMatrix2KGML(mat, pathname, nodePrefix="n", overwrite=FALSE, ..., verbose=FALSE)
```

Arguments

mat	A matrix, interpreted of the adjacency matrix of a graph.
pathname	The full path name of the XML file to be written.
nodePrefix	A character value giving the prefix to which the node index in 'mat' will be appended.
overwrite	If TRUE and file already exists, overwrite it.
...	Further arguments to be passed to plotKEGGgraph.
verbose	If TRUE, extra information is output.

Value

None.

Author(s)

Laurent Jacob, Pierre Neuvial and Sandrine Dudoit

See Also

parseKGML2Graph

Examples

```
library("KEGGgraph")
library("rrcov")

## Create a random graph
graph <- randomWAMGraph(nnodes=5, nedges=7, verbose=TRUE)
plot(graph)

## Retrieve its adjacency matrix
A <- graph@adjMat

## write it to KGML file
grPathname <- "randomWAMGraph.xml"
writeAdjacencyMatrix2KGML(A, pathname=grPathname, verbose=TRUE, overwrite=TRUE)

## read it from file
gr <- parseKGML2Graph(grPathname)

## Two examples of Laplacians from the same graph
lapMI <- laplacianFromA(A, ltype="meanInfluence")
print(lapMI)

lapN <- laplacianFromA(A, ltype="normalized")
print(lapN)

U <- lapN$U
p <- nrow(A)
sigma <- diag(p)/sqrt(p)
```

```
X <- twoSampleFromGraph(100, 120, shiftM2=1, sigma, U=U, k=3)

## T2
t <- T2.test(X$X1,X$X2)
str(t)

tu <- graph.T2.test(X$X1, X$X2, lfA=lapMI, k=3)
str(tu)
```

Index

∗ **datasets**
annLoi2008, 4
classLoi2008, 7
exprLoi2008, 8
grListKEGG, 14

AN.test, 2, *5*, *15*
annLoi2008, 4

BS.test, *3*, 5, *15*

character, *10, 11*, *15*, *17, 18*, *21*, *27*
classLoi2008, 7
connectedComp, *9*

exprLoi2008, 8

FALSE, *18*

getConnectedComponentList, 9, *22*
getKEGGPathways, 10
getSignedGraph, 11
graph, *9–11*, *18*, *21–23*
graph.T2.test, *3*, *5*, 12, *15*
graphAM, *13*, *20*
graphNEL, *13*
grListKEGG, 14

hyper.test, *3*, *5*, 14

integer, *15*, *25*

KEGGpathway2Graph, *10*

laplacianFromA, *13*, 16, *21*
list, *2*, *5*, *9, 10*, *13*, *15*, *17, 18*, *22, 23*, *25*
logical, *2*, *5*, *18*, *23*

matrix, *2*, *5*, *13*, *21*, *27*

NA, *2*, *5*
numeric, *2*, *5*, *13*, *15*, *18*, *20–22*, *25*

parseKGML, *10*
parseKGML2Graph, *27*
plot, *19*
plotKEGGgraph, *19*
plotValuedGraph, 18

randomWAMGraph, 20

T2.test, *13*
testOneConnectedComponent, 21, *23*
testOneGraph, *22*, 23
TRUE, *9–11*, *15*, *18*, *20*, *22, 23*, *27*
twoSampleFromGraph, 25

vector, *2*, *11*, *18*, *21*

writeAdjacencyMatrix2KGML, 26