Package 'gMCPLite'

July 22, 2025

Title Lightweight Graph Based Multiple Comparison Procedures

Version 0.1.5

Description A lightweight fork of 'gMCP' with functions for graphical
described multiple test procedures introduced in
Bretz et al. (2009) <doi:10.1002/sim.3495> and
Bretz et al. (2011) <doi:10.1002/bimj.201000239>.
Implements a flexible function using 'ggplot2' to create
multiplicity graph visualizations.
Contains instructions of multiplicity graph and graphical testing for
group sequential design, described in
Maurer and Bretz (2013) <doi:10.1080/19466315.2013.807748>,
with necessary unit testing using 'testthat'.

License GPL-3

URL `https://merck.github.io/gMCPLite/`,
`https://github.com/Merck/gMCPLite`

BugReports `https://github.com/Merck/gMCPLite/issues`

Encoding UTF-8

VignetteBuilder knitr

Depends R (>= 3.6.0)

Imports ggplot2, graphics, grDevices, grid, MASS, methods, mvtnorm,
stats, utils

Suggests covr, dplyr, gridExtra, gsDesign, gt, kableExtra, knitr,
multcomp, ragg, rmarkdown, scales, testthat (>= 3.0.0), tibble

Config/testthat/edition 3

RoxygenNote 7.2.3

NeedsCompilation yes

Author Yalin Zhu [aut] (ORCID: <`https://orcid.org/0000-0003-3830-8660`>),
Yilong Zhang [aut],
Xuan Deng [aut],
Keaven Anderson [aut],
Nan Xiao [aut, cre] (ORCID: <`https://orcid.org/0000-0002-0250-5673`>),

Kornelius Rohmeyer [ctb] (gMCP author),
Florian Klinglmueller [ctb] (gMCP author),
gMCP project contributors [cph] (gMCP package),
Merck & Co., Inc., Rahway, NJ, USA and its affiliates [cph]

Maintainer Nan Xiao <nan.xiao1@merck.com>

Repository CRAN

Date/Publication 2024-01-11 19:30:02 UTC

Contents

`bdiagNA` *Create a Block Diagonal Matrix with NA outside the diagonal*

Description

Build a block diagonal matrix with NA values outside the diagonal given several building block matrices.

Usage

```
bdiagNA(...)
```

Arguments

`...`	individual matrices or a `list` of matrices.

Details

This function is useful to build the correlation matrices, when only partial knowledge of the correlation exists.

Value

A block diagonal matrix with NA values outside the diagonal.

Author(s)

Kornelius Rohmeyer `<rohmeyer@small-projects.de>`

See Also

`gMCP`

Examples

```
bdiagNA(diag(3), matrix(1/2,nr=3,nc=3), diag(2))
```

`bonferroni.test` *Weighted Bonferroni-test*

Description

Weighted Bonferroni-test

Usage

```
bonferroni.test(
  pvalues,
  weights,
  alpha = 0.05,
  adjPValues = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`pvalues`	A numeric vector specifying the p-values.
`weights`	A numeric vector of weights.
`alpha`	A numeric specifying the maximal allowed type one error rate. If `adjPValues==TRUE` (default) the parameter `alpha` is not used.
`adjPValues`	Logical scalar. If `TRUE` (the default) an adjusted p-value for the weighted Bonferroni-test is returned. Otherwise if `adjPValues==FALSE` a logical value is returned whether the null hypothesis can be rejected.
`verbose`	Logical scalar. If `TRUE` verbose output is generated.
`...`	Further arguments possibly passed by `gMCP` which will be used by other test procedures but not this one.

Value

adjusted p-value or decision of rejection

Examples

```
bonferroni.test(pvalues=c(0.1,0.2,0.05), weights=c(0.5,0.5,0))
bonferroni.test(pvalues=c(0.1,0.2,0.05), weights=c(0.5,0.5,0), adjPValues=FALSE)
```

bonferroni.trimmed.simes.test

Trimmed Simes test for intersections of two hypotheses and otherwise weighted Bonferroni-test

Description

Trimmed Simes test for intersections of two hypotheses and otherwise weighted Bonferroni-test

Usage

```
bonferroni.trimmed.simes.test(
  pvalues,
  weights,
  alpha = 0.05,
  adjPValues = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`pvalues`	A numeric vector specifying the p-values.
`weights`	A numeric vector of weights.
`alpha`	A numeric specifying the maximal allowed type one error rate. If `adjPValues==TRUE` (default) the parameter `alpha` is not used.
`adjPValues`	Logical scalar. If `TRUE` (the default) an adjusted p-value for the weighted test is returned. Otherwise if `adjPValues==FALSE` a logical value is returned whether the null hypothesis can be rejected.
`verbose`	Logical scalar. If `TRUE` verbose output is generated.
`...`	Further arguments possibly passed by `gMCP` which will be used by other test procedures but not this one.

Value

adjusted p-value or decision of rejection

References

Brannath, W., Bretz, F., Maurer, W., & Sarkar, S. (2009). Trimmed Weighted Simes Test for Two One-Sided Hypotheses With Arbitrarily Correlated Test Statistics. Biometrical Journal, 51(6), 885-898.

Examples

```
bonferroni.trimmed.simes.test(pvalues=c(0.1,0.2,0.05), weights=c(0.5,0.5,0))
bonferroni.trimmed.simes.test(pvalues=c(0.1,0.2,0.05), weights=c(0.5,0.5,0), adjPValues=FALSE)
```

---

checkCorrelation *Check correlation matrix*

Description

Sanity checks for the correlation matrix.

Usage

```
checkCorrelation(m, returnMessage = FALSE, na.allowed = TRUE)
```

Arguments

m	TBA
returnMessage	TBA
na.allowed	TBA

Details

Checks the following properties:

- Values must be between -1 and 1.
- Diagonal must be equal to 1.
- Matrix must be symmetric.

Value

Logical

Examples

```
NULL
```

`entangledMCP-class` *Class entangledMCP*

Description

A entangledMCP object describes ... TODO

Slots

`subgraphs` A list of graphs of class graphMCP.

`weights` A numeric.

`graphAttr` A list for graph attributes like color, etc.

Methods

print `signature(object = "entangledMCP")`: A method for printing the data of the entangled graph to the R console.

getMatrices `signature(object = "entangledMCP")`: A method for getting the list of transition matrices of the entangled graph.

getWeights `signature(object = "entangledMCP")`: A method for getting the matrix of weights of the entangled graph.

getRejected `signature(object = "entangledMCP")`: A method for getting the information whether the hypotheses are marked in the graph as already rejected. If a second optional argument `node` is specified, only for these nodes the boolean vector will be returned.

getXCoordinates `signature(object = "entangledMCP")`: A method for getting the x coordinates of the graph. If a second optional argument `node` is specified, only for these nodes the x coordinates will be returned. If x coordinates are not yet set, `NULL` is returned.

getYCoordinates `signature(object = "entangledMCP")`: A method for getting the y coordinates of the graph If a second optional argument `node` is specified, only for these nodes the x coordinates will be returned. If y coordinates are not yet set, `NULL` is returned.

Author(s)

Kornelius Rohmeyer `<rohmeyer@small-projects.de>`

See Also

`graphMCP`

Examples

```
g1 <- BonferroniHolm(2)
g2 <- BonferroniHolm(2)

graph <- new("entangledMCP", subgraphs=list(g1,g2), weights=c(0.5,0.5))
```

```
getMatrices(graph)
getWeights(graph)
```

exampleGraphs *Functions that create different example graphs*

Description

Functions that creates example graphs, e.g. graphs that represents a Bonferroni-Holm adjustment, parallel gatekeeping or special procedures from selected papers.

Usage

```
BonferroniHolm(n, weights = rep(1/n, n))

BretzEtAl2011()

BauerEtAl2001()

BretzEtAl2009a()

BretzEtAl2009b()

BretzEtAl2009c()

HommelEtAl2007()

HommelEtAl2007Simple()

parallelGatekeeping()

improvedParallelGatekeeping()

fallback(weights)

fixedSequence(n)

simpleSuccessiveI()

simpleSuccessiveII()

truncatedHolm(gamma)

generalSuccessive(weights = c(1/2, 1/2), gamma, delta)

HuqueAloshEtBhore2011()
```

```
HungEtWang2010(nu, tau, omega)

MaurerEtAl1995()

cycleGraph(nodes, weights)

improvedFallbackI(weights = rep(1/3, 3))

improvedFallbackII(weights = rep(1/3, 3))

FerberTimeDose2011(times, doses, w = "\\nu")

Ferber2011(w)

Entangled1Maurer2012()

Entangled2Maurer2012()

WangTing2014(nu, tau)
```

Arguments

`n`	Number of hypotheses.
`weights`	Numeric vector of node weights.
`gamma`	An optional number in [0,1] specifying the value for variable gamma.
`delta`	An optional number in [0,1] specifying the value for variable delta.
`nu`	An optional number in [0,1] specifying the value for variable nu.
`tau`	An optional number in [0,1] specifying the value for variable tau.
`omega`	An optional number in [0,1] specifying the value for variable omega.
`nodes`	Character vector of node names.
`times`	Number of time points.
`doses`	Number of dose levels.
`w`	Further variable weight(s) in graph.

Details

We are providing functions and not the resulting graphs directly because this way you have additional examples: You can look at the function body with `body` and see how the graph is built.

list("BonferroniHolm") Returns a graph that represents a Bonferroni-Holm adjustment. The result is a complete graph, where all nodes have the same weights and each edge weight is $\frac{1}{n-1}$.

list("BretzEtAl2011") Graph in figure 2 from Bretz et al. See references (Bretz et al. 2011).

list("HommelEtAl2007") Graph from Hommel et al. See references (Hommel et al. 2007).

list("parallelGatekeeping") Graph for parallel gatekeeping. See references (Dmitrienko et al. 2003).

list("improvedParallelGatekeeping") Graph for improved parallel gatekeeping. See references (Hommel et al. 2007).

list("HungEtWang2010") Graph from Hung et Wang. See references (Hung et Wang 2010).

list("MaurerEtAl1995") Graph from Maurer et al. See references (Maurer et al. 1995).

list("cycleGraph") Cycle graph. The weight `weights[i]` specifies the edge weight from node i to node $i + 1$ for $i = 1, \ldots, n - 1$ and `weight[n]` from node n to node 1.

list("improvedFallbackI") Graph for the improved Fallback Procedure by Wiens & Dmitrienko. See references (Wiens et Dmitrienko 2005).

list("improvedFallbackII") Graph for the improved Fallback Procedure by Hommel & Bretz. See references (Hommel et Bretz 2008).

list("Ferber2011") Graph from Ferber et al. See references (Ferber et al. 2011).

list("FerberTimeDose2011") Graph from Ferber et al. See references (Ferber et al. 2011).

list("Entangled1Maurer2012") Entangled graph from Maurer et al. TODO: Add references as soon as they are available.

Value

A graph of class `graphMCP` that represents a sequentially rejective multiple test procedure.

Author(s)

Kornelius Rohmeyer `<rohmeyer@small-projects.de>`

References

Holm, S. (1979). A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6, 65-70.

Dmitrienko, A., Offen, W., Westfall, P.H. (2003). Gatekeeping strategies for clinical trials that do not require all primary effects to be significant. Statistics in Medicine. 22, 2387-2400.

Bretz, F., Maurer, W., Brannath, W., Posch, M.: A graphical approach to sequentially rejective multiple test procedures. Statistics in Medicine 2009 vol. 28 issue 4 page 586-604. https://www.meduniwien.ac.at/fwf_adaptive/papers/bretz_2009_22.pdf

Bretz, F., Maurer, W. and Hommel, G. (2011), Test and power considerations for multiple endpoint analyses using sequentially rejective graphical procedures. Statistics in Medicine, 30: 1489–1501.

Hommel, G., Bretz, F. und Maurer, W. (2007). Powerful short-cuts for multiple testing procedures with special reference to gatekeeping strategies. Statistics in Medicine, 26(22), 4063-4073.

Hommel, G., Bretz, F. (2008): Aesthetics and power considerations in multiple testing - a contradiction? Biometrical Journal 50:657-666.

Hung H.M.J., Wang S.-J. (2010). Challenges to multiple testing in clinical trials. Biometrical Journal 52, 747-756.

W. Maurer, L. Hothorn, W. Lehmacher: Multiple comparisons in drug clinical trials and preclinical assays: a-priori ordered hypotheses. In Biometrie in der chemisch-pharmazeutischen Industrie, Vollmar J (ed.). Fischer Verlag: Stuttgart, 1995; 3-18.

Maurer, W., & Bretz, F. (2013). Memory and other properties of multiple test procedures generated by entangled graphs. Statistics in medicine, 32 (10), 1739-1753.

Wiens, B.L., Dmitrienko, A. (2005): The fallback procedure for evaluating a single family of hypotheses. Journal of Biopharmaceutical Statistics 15:929-942.

Wang, B., Ting, N. (2014). An Application of Graphical Approach to Construct Multiple Testing Procedures in a Hypothetical Phase III Design. Frontiers in public health, 1 (75).

Ferber, G. Staner, L. and Boeijinga, P. (2011): Structured multiplicity and confirmatory statistical analyses in pharmacodynamic studies using the quantitative electroencephalogram, Journal of neuroscience methods, Volume 201, Issue 1, Pages 204-212.

Examples

```
g <- BonferroniHolm(5)

gMCP(g, pvalues=c(0.1, 0.2, 0.4, 0.4, 0.7))

HungEtWang2010()
HungEtWang2010(nu=1)
```

`generatePvals` *generatePvals*

Description

compute adjusted p-values either for the closed test defined by the graph or for each elementary hypotheses within each intersection hypotheses

Usage

```
generatePvals(
  g,
  w,
  cr,
  p,
  adjusted = TRUE,
  hint = generateWeights(g, w),
  upscale = FALSE
)
```

Arguments

`g`	graph defined as a matrix, each element defines how much of the local alpha reserved for the hypothesis corresponding to its row index is passed on to the hypothesis corresponding to its column index
`w`	vector of weights, defines how much of the overall alpha is initially reserved for each elementary hypothesis

`cr`	correlation matrix if p-values arise from one-sided tests with multivariate normal distributed test statistics for which the correlation is partially known. Unknown values can be set to NA. (See details for more information)
`p`	vector of observed unadjusted p-values, that belong to test-statistics with a joint multivariate normal null distribution with (partially) known correlation matrix `cr`
`adjusted`	logical, if TRUE (default) adjusted p-values for the closed test are returned, else a matrix of p-values adjusted only for each intersection hypothesis is returned
`hint`	if intersection hypotheses weights have already been computed (output of `generateWeights`) can be passed here otherwise will be computed during execution
`upscale`	if `FALSE` (default) the p-values are additionally adjusted for the case that non-exhaustive weights are specified. (See details)

Details

It is assumed that under the global null hypothesis $(\Phi^{-1}(1-p_1),...,\Phi^{-1}(1-p_m))$ follow a multivariate normal distribution with correlation matrix `cr` where Φ^{-1} denotes the inverse of the standard normal distribution function.

For example, this is the case if $p_1,...,p_m$ are the raw p-values from one-sided z-tests for each of the elementary hypotheses where the correlation between z-test statistics is generated by an overlap in the observations (e.g. comparison with a common control, group-sequential analyses etc.). An application of the transformation $\Phi^{-1}(1-p_i)$ to raw p-values from a two-sided test will not in general lead to a multivariate normal distribution. Partial knowledge of the correlation matrix is supported. The correlation matrix has to be passed as a numeric matrix with elements of the form: $cr[i,i] = 1$ for diagonal elements, $cr[i,j] = \rho_{ij}$, where ρ_{ij} is the known value of the correlation between $\Phi^{-1}(1-p_i)$ and $\Phi^{-1}(1-p_j)$ or `NA` if the corresponding correlation is unknown. For example cr[1,2]=0 indicates that the first and second test statistic are uncorrelated, whereas cr[2,3] = NA means that the true correlation between statistics two and three is unknown and may take values between -1 and 1. The correlation has to be specified for complete blocks (ie.: if cor(i,j), and cor(i,k) for i!=j!=k are specified then cor(j,k) has to be specified as well) otherwise the corresponding intersection null hypotheses tests are not uniquely defined and an error is returned.

The parametric tests in (Bretz et al. (2011)) are defined such that the tests of intersection null hypotheses always exhaust the full alpha level even if the sum of weights is strictly smaller than one. This has the consequence that certain test procedures that do not test each intersection null hypothesis at the full level alpha may not be implemented (e.g., a single step Dunnett test). If `upscale` is set to `FALSE` (default) the parametric tests are performed at a reduced level alpha of sum(w) * alpha and p-values adjusted accordingly such that test procedures with non-exhaustive weighting strategies may be implemented. If set to `TRUE` the tests are performed as defined in Equation (3) of (Bretz et al. (2011)).

Value

If adjusted is set to true returns a vector of adjusted p-values. Any elementary null hypothesis is rejected if its corresponding adjusted p-value is below the predetermined alpha level. For adjusted set to false a matrix with p-values adjusted only within each intersection hypotheses is returned. The intersection corresponding to each line is given by conversion of the line number into binary (eg. 13 is binary 1101 and corresponds to (H1,H2,H4)). If any adjusted p-value within a given line falls below alpha, then the corresponding intersection hypotheses can be rejected.

Author(s)

Florian Klinglmueller

References

Bretz F, Maurer W, Brannath W, Posch M; (2008) - A graphical approach to sequentially rejective multiple testing procedures. - Stat Med - 28/4, 586-604 Bretz F, Posch M, Glimm E, Klinglmueller F, Maurer W, Rohmeyer K; (2011) - Graphical approaches for multiple endpoint problems using weighted Bonferroni, Simes or parametric tests - to appear

Examples

```
## Define some graph as matrix
g <- matrix(c(0,0,1,0, 0,0,0,1, 0,1,0,0, 1,0,0,0), nrow = 4, byrow=TRUE)
## Choose weights
w <- c(.5,.5,0,0)
## Some correlation (upper and lower first diagonal 1/2)
c <- diag(4)
c[1:2,3:4] <- NA
c[3:4,1:2] <- NA
c[1,2] <- 1/2
c[2,1] <- 1/2
c[3,4] <- 1/2
c[4,3] <- 1/2
## p-values as Section 3 of Bretz et al. (2011),
p <- c(0.0121,0.0337,0.0084,0.0160)

## Boundaries for correlated test statistics at alpha level .05:
generatePvals(g,w,c,p)

g <- Entangled2Maurer2012()
generatePvals(g=g, cr=diag(5), p=rep(0.1,5))
```

`generateWeights` *generateWeights*

Description

compute Weights for each intersection Hypotheses in the closure of a graph based multiple testing procedure

Usage

```
generateWeights(g, w)
```

Arguments

`g`	Graph either defined as a matrix (each element defines how much of the local alpha reserved for the hypothesis corresponding to its row index is passed on to the hypothesis corresponding to its column index), as `graphMCP` object or as `entangledMCP` object.
`w`	Vector of weights, defines how much of the overall alpha is initially reserved for each elementary hypothesis. Can be missing if `g` is a `graphMCP` object (in which case the weights from the graph object are used). Will be ignored if `g` is an `entangledMCP` object (since then the matrix of weights from this object is used).

Value

Returns matrix with each row corresponding to one intersection hypothesis in the closure of the multiple testing problem. The first half of elements indicate whether an elementary hypotheses is in the intersection (1) or not (0). The second half of each row gives the weights allocated to each elementary hypotheses in the intersection.

Author(s)

Florian Klinglmueller <float@lefant.net>, Kornelius Rohmeyer <`rohmeyer@small-projects.de`>

References

Bretz F, Maurer W, Brannath W, Posch M; (2008) - A graphical approach to sequentially rejective multiple testing procedures. - Stat Med - 28/4, 586-604 Bretz F, Posch M, Glimm E, Klinglmueller F, Maurer W, Rohmeyer K; (2011) - Graphical approaches for multiple endpoint problems using weighted Bonferroni, Simes or parametric tests - to appear

Examples

```
 g <- matrix(c(0,0,1,0,
              0,0,0,1,
              0,1,0,0,
              1,0,0,0), nrow = 4,byrow=TRUE)
 ## Choose weights
 w <- c(.5,.5,0,0)
 ## Weights of conventional gMCP test:
 generateWeights(g,w)

g <- Entangled2Maurer2012()
generateWeights(g)
```

gMCP *Graph based Multiple Comparison Procedures*

Description

Performs a graph based multiple test procedure for a given graph and unadjusted p-values.

Usage

```
gMCP(
  graph,
  pvalues,
  test,
  correlation,
  alpha = 0.05,
  approxEps = TRUE,
  eps = 10^(-3),
  ...,
 upscale = ifelse(missing(test) && !missing(correlation) || !missing(test) && test ==
    "Bretz2011", TRUE, FALSE),
  useC = FALSE,
  verbose = FALSE,
  keepWeights = FALSE,
  adjPValues = TRUE
)
```

Arguments

`graph`	A graph of class `graphMCP`.
`pvalues`	A numeric vector specifying the p-values for the graph based MCP. Note the assumptions in the details section for the parametric tests, when a correlation is specified.
`test`	Should be either `"Bonferroni"`, `"Simes"` or `"parametric"`. If not specified by default the Bonferroni-based test procedure is used if no correlation is specified or the algorithm from Bretz et al. 2011 if a correlation is specified. If `test` is set to `"Simes"` the weighted Simes test will be performed for each subset of hypotheses.
`correlation`	Optional correlation matrix. If the weighted Simes test is performed, it is checked whether type I error rate can be ensured and a warning is given if this is not the case. For parametric tests the p-values must arise from one-sided tests with multivariate normal distributed test statistics for which the correlation is (partially) known. In that case a weighted parametric closed test is performed (also see `generatePvals`). Unknown values can be set to NA. (See details for more information)
`alpha`	A numeric specifying the maximal allowed type one error rate.

`approxEps`	A boolean specifying whether epsilon values should be substituted with the value given in the parameter `eps`.
`eps`	A numeric scalar specifying a value for epsilon edges.
`...`	Test specific arguments can be given here.
`upscale`	Logical. If `upscale=FALSE` then for each intersection of hypotheses (i.e. each subgraph) a weighted test is performed at the possibly reduced level alpha of sum(w)*alpha, where sum(w) is the sum of all node weights in this subset. If `upscale=TRUE` all weights are upscaled, so that sum(w)=1. For backward compatibility the default value is TRUE if a the parameter `test` is missing, but parameter `correlation` is specified or if `test=="Bretz2011"`.
`useC`	Logical scalar. If `TRUE` neither adjusted p-values nor intermediate graphs are returned, but the calculation is sped up by using code written in C. THIS CODE IS NOT FOR PRODUCTIVE USE YET! If approxEps is `FALSE` and the graph contains epsilon edges, a warning is thrown and `useC` will be ignored.
`verbose`	Logical scalar. If `TRUE` verbose output is generated.
`keepWeights`	Logical scalar. If `FALSE` the weight of a node without outgoing edges is set to 0 if it is removed. Otherwise it keeps its weight.
`adjPValues`	Logical scalar. If `FALSE` no adjusted p-values will be calculated. Especially for the weighted Simes test this will result in significantly less calculations in most cases.

Details

For the Bonferroni procedure the p-values can arise from any statistical test, but if you improve the test by specifying a correlation matrix, the following assumptions apply:

It is assumed that under the global null hypothesis $(\Phi^{-1}(1-p_1), ..., \Phi^{-1}(1-p_m))$ follow a multivariate normal distribution with correlation matrix `correlation` where Φ^{-1} denotes the inverse of the standard normal distribution function.

For example, this is the case if $p_1, ..., p_m$ are the raw p-values from one-sided z-tests for each of the elementary hypotheses where the correlation between z-test statistics is generated by an overlap in the observations (e.g. comparison with a common control, group-sequential analyses etc.). An application of the transformation $\Phi^{-1}(1-p_i)$ to raw p-values from a two-sided test will not in general lead to a multivariate normal distribution. Partial knowledge of the correlation matrix is supported. The correlation matrix has to be passed as a numeric matrix with elements of the form: $correlation[i,i] = 1$ for diagonal elements, $correlation[i,j] = \rho_{ij}$, where ρ_{ij} is the known value of the correlation between $\Phi^{-1}(1-p_i)$ and $\Phi^{-1}(1-p_j)$ or `NA` if the corresponding correlation is unknown. For example correlation[1,2]=0 indicates that the first and second test statistic are uncorrelated, whereas correlation[2,3] = NA means that the true correlation between statistics two and three is unknown and may take values between -1 and 1. The correlation has to be specified for complete blocks (ie.: if cor(i,j), and cor(i,j') for i!=j!=j' are specified then cor(j,j') has to be specified as well) otherwise the corresponding intersection null hypotheses tests are not uniquely defined and an error is returned.

For further details see the given references.

Value

An object of class `gMCPResult`, more specifically a list with elements

`graphs` list of graphs

`pvalues` p-values

`rejected` logical whether hypotheses could be rejected

`adjPValues` adjusted p-values

Author(s)

Kornelius Rohmeyer <`rohmeyer@small-projects.de`>

References

Frank Bretz, Willi Maurer, Werner Brannath, Martin Posch: A graphical approach to sequentially rejective multiple test procedures. Statistics in Medicine 2009 vol. 28 issue 4 page 586-604. `https://www.meduniwien.ac.at/fwf_adaptive/papers/bretz_2009_22.pdf`

Bretz F., Posch M., Glimm E., Klinglmueller F., Maurer W., Rohmeyer K. (2011): Graphical approaches for multiple endpoint problems using weighted Bonferroni, Simes or parametric tests. Biometrical Journal 53 (6), pages 894-913, Wiley. doi:10.1002/bimj.201000239

Strassburger K., Bretz F.: Compatible simultaneous lower confidence bounds for the Holm procedure and other Bonferroni based closed tests. Statistics in Medicine 2008; 27:4914-4927.

Hommel G., Bretz F., Maurer W.: Powerful short-cuts for multiple testing procedures with special reference to gatekeeping strategies. Statistics in Medicine 2007; 26:4063-4073.

Guilbaud O.: Simultaneous confidence regions corresponding to Holm's stepdown procedure and other closed-testing procedures. Biometrical Journal 2008; 50:678-692.

See Also

`graphMCP` `multcomp::contrMat()`

Examples

```
g <- BonferroniHolm(5)
gMCP(g, pvalues=c(0.01, 0.02, 0.04, 0.04, 0.7))
# Simple Bonferroni with empty graph:
g2 <- matrix2graph(matrix(0, nrow=5, ncol=5))
gMCP(g2, pvalues=c(0.01, 0.02, 0.04, 0.04, 0.7))
# With 'upscale=TRUE' equal to BonferroniHolm:
gMCP(g2, pvalues=c(0.01, 0.02, 0.04, 0.04, 0.7), upscale=TRUE)
```

`gMCP.extended` *Graph based Multiple Comparison Procedures*

Description

Performs a graph based multiple test procedure for a given graph and unadjusted p-values.

Usage

```
gMCP.extended(
  graph,
  pvalues,
  test,
  alpha = 0.05,
  eps = 10^(-3),
  upscale = FALSE,
  verbose = FALSE,
  adjPValues = TRUE,
  ...
)
```

Arguments

`graph`	A graph of class `graphMCP`.
`pvalues`	A numeric vector specifying the p-values for the graph based MCP. Note the assumptions in the description of the selected test (if there are any - for example `test=bonferroni.test` has no further assumptions, but `test=parametric.test` assumes p-values from a multivariate normal distribution).
`test`	A weighted test function. The package gMCP provides the following weighted test functions: **bonferroni.test** Bonferroni test - see `?bonferroni.test` for details. **parametric.test** Parametric test - see `?parametric.test` for details. **simes.test** Simes test - see `?simes.test` for details. **bonferroni.trimmed.simes.test** Trimmed Simes test for intersections of two hypotheses and otherwise Bonferroni - see `?bonferroni.trimmed.simes.test` for details. **simes.on.subsets.test** Simes test for intersections of hypotheses from certain sets and otherwise Bonferroni - see `?simes.on.subsets.test` for details. To provide your own test function see `?weighted.test.function`.
`alpha`	A numeric specifying the maximal allowed type one error rate.
`eps`	A numeric scalar specifying a value for epsilon edges.
`upscale`	Logical. If `upscale=FALSE` then for each intersection of hypotheses (i.e. each subgraph) a weighted test is performed at the possibly reduced level alpha of sum(w)*alpha, where sum(w) is the sum of all node weights in this subset. If `upscale=TRUE` all weights are upscaled, so that sum(w)=1.

`verbose`	Logical scalar. If `TRUE` verbose output is generated during sequentially rejection steps.
`adjPValues`	Logical scalar. If `FALSE` no adjusted p-values will be calculated. Especially for the weighted Simes test this will result in significantly less calculations in most cases.
`...`	Test specific arguments can be given here.

Value

An object of class `gMCPResult`, more specifically a list with elements

`graphs` list of graphs

`pvalues` p-values

`rejected` logical whether hypotheses could be rejected

`adjPValues` adjusted p-values

Author(s)

Kornelius Rohmeyer `<rohmeyer@small-projects.de>`

References

Frank Bretz, Willi Maurer, Werner Brannath, Martin Posch: A graphical approach to sequentially rejective multiple test procedures. Statistics in Medicine 2009 vol. 28 issue 4 page 586-604. `https://www.meduniwien.ac.at/fwf_adaptive/papers/bretz_2009_22.pdf`

Bretz F., Posch M., Glimm E., Klinglmueller F., Maurer W., Rohmeyer K. (2011): Graphical approaches for multiple endpoint problems using weighted Bonferroni, Simes or parametric tests. Biometrical Journal 53 (6), pages 894-913, Wiley. doi:10.1002/bimj.201000239

Strassburger K., Bretz F.: Compatible simultaneous lower confidence bounds for the Holm procedure and other Bonferroni based closed tests. Statistics in Medicine 2008; 27:4914-4927.

Hommel G., Bretz F., Maurer W.: Powerful short-cuts for multiple testing procedures with special reference to gatekeeping strategies. Statistics in Medicine 2007; 26:4063-4073.

Guilbaud O.: Simultaneous confidence regions corresponding to Holm's stepdown procedure and other closed-testing procedures. Biometrical Journal 2008; 50:678-692.

See Also

`graphMCP` `multcomp::contrMat()`

Examples

```
g <- BonferroniHolm(5)
gMCP(g, pvalues=c(0.01, 0.02, 0.04, 0.04, 0.7))
# Simple Bonferroni with empty graph:
g2 <- matrix2graph(matrix(0, nrow=5, ncol=5))
gMCP(g2, pvalues=c(0.01, 0.02, 0.04, 0.04, 0.7))
# With 'upscale=TRUE' equal to BonferroniHolm:
gMCP(g2, pvalues=c(0.01, 0.02, 0.04, 0.04, 0.7), upscale=TRUE)
```

```
# Entangled graphs:
g3 <- Entangled2Maurer2012()
gMCP(g3, pvalues=c(0.01, 0.02, 0.04, 0.04, 0.7), correlation=diag(5))
```

`gMCPResult-class` *Class gMCPResult*

Description

A gMCPResult object describes an evaluated sequentially rejective multiple test procedure.

Slots

`graphs` Object of class `list`.

`alpha` A `numeric` specifying the maximal type I error rate.

`pvalues` The `numeric` vector of p-values.

`rejected` The `logical` vector of rejected null hypotheses.

`adjPValues` The `numeric` vector of adjusted p-values.

Author(s)

Kornelius Rohmeyer `<rohmeyer@small-projects.de>`

See Also

`gMCP`

`gPADInterim-class` *Class gPADInterim*

Description

A gPADInterim object describes an object holding interim information for an adaptive procedure that is based on a preplanned graphical procedure.

Slots

`Aj` Object of class `numeric`. Giving partial conditional errors (PCEs) for all elementary hypotheses in each intersection hypothesis

`BJ` A `numeric` specifying the sum of PCEs per intersection hypothesis.

`z1` The `numeric` vector of first stage z-scores.

`v` A `numeric` specifying the proportion of measurements collected up to interim

`preplanned` Object of class `graphMCP` specifying the preplanned graphical procedure.

`alpha` A `numeric` giving the alpha level of the pre-planned test

Author(s)

Florian Klinglmueller <`float@lefant.net`>

See Also

`gMCP`

`graphMCP-class` *Class graphMCP*

Description

A graphMCP object describes a sequentially rejective multiple test procedure.

Slots

`m` A transition matrix. Can be either `numerical` or `character` depending whether the matrix contains variables or not. Row and column names will be the names of the nodes.

`weights` A numeric.

`edgeAttr` A list for edge attributes.

`nodeAttr` A list for node attributes.

Methods

getMatrix `signature(object = "graphMCP")`: A method for getting the transition matrix of the graph.

getWeights `signature(object = "graphMCP")`: A method for getting the weights. If a third optional argument `node` is specified, only for these nodes the weight will be returned.

setWeights `signature(object = "graphMCP")`: A method for setting the weights. If a third optional argument `node` is specified, only for these nodes the weight will be set.

getRejected `signature(object = "graphMCP")`: A method for getting the information whether the hypotheses are marked in the graph as already rejected. If a second optional argument `node` is specified, only for these nodes the boolean vector will be returned.

getXCoordinates `signature(object = "graphMCP")`: A method for getting the x coordinates of the graph. If a second optional argument `node` is specified, only for these nodes the x coordinates will be returned. If x coordinates are not set yet `NULL` is returned.

getYCoordinates `signature(object = "graphMCP")`: A method for getting the y coordinates of the graph If a second optional argument `node` is specified, only for these nodes the x coordinates will be returned. If y coordinates are not set yet `NULL` is returned.

setEdge `signature(from="character", to="character", graph="graphNEL", weights="numeric")`: A method for adding new edges with the given weights.

setEdge `signature(from="character", to="character", graph="graphMCP", weights="character")`: A method for adding new edges with the given weights.

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

Examples

```
m <- rbind(H11=c(0,   0.5, 0,   0.5, 0,   0  ),
H21=c(1/3, 0,   1/3, 0,   1/3, 0  ),
H31=c(0,   0.5, 0,   0,   0,   0.5),
H12=c(0,   1,   0,   0,   0,   0  ),
H22=c(0.5, 0,   0.5, 0,   0,   0  ),
H32=c(0,   1,   0,   0,   0,   0  ))

weights <- c(1/3, 1/3, 1/3, 0, 0, 0)

# Graph creation
graph <- new("graphMCP", m=m, weights=weights)

# Visualization settings
nodeX <- rep(c(100, 300, 500), 2)
nodeY <- rep(c(100, 300), each=3)
graph@nodeAttr$X <- nodeX
graph@nodeAttr$Y <- nodeY

getWeights(graph)

getRejected(graph)

pvalues <- c(0.1, 0.008, 0.005, 0.15, 0.04, 0.006)
result <- gMCP(graph, pvalues)

getWeights(result@graphs[[4]])
getRejected(result@graphs[[4]])
```

---

graphTest *Multiple testing using graphs*

Description

Implements the graphical test procedure described in Bretz et al. (2009). Note that the gMCP function in the gMCP package performs the same task.

Usage

```
graphTest(
  pvalues,
  weights = NULL,
  alpha = 0.05,
  G = NULL,
  cr = NULL,
  graph = NULL,
  verbose = FALSE,
  test,
  upscale = FALSE
)
```

Arguments

`pvalues`	Either a vector or a matrix containing the local p-values for the hypotheses in the rows.
`weights`	Initial weight levels for the test procedure, in case of multiple graphs this needs to be a matrix.
`alpha`	Overall alpha level of the procedure. For entangled graphs `alpha` should be a numeric vector of length equal to the number of graphs, each element specifying the partial alpha for the respective graph. The overall alpha level equals `sum(alpha)`.
`G`	For simple graphs `G` should be a numeric matrix determining the graph underlying the test procedure. Note that the diagonal need to contain only 0s, while the rows need to sum to 1. For entangled graphs it needs to be a list containing the different graph matrices as elements.
`cr`	Correlation matrix that should be used for the parametric test. If `cr==NULL` the Bonferroni based test procedure is used.
`graph`	As an alternative to the specification via `weights` and `G` one can also hand over a `graphMCP` object to the code. `graphMCP` objects can be created for example with the `graphGUI` function.
`verbose`	If verbose is TRUE, additional information about the graphical rejection procedure is displayed.
`test`	In the parametric case there is more than one way to handle subgraphs with less than the full alpha. If the parameter `test` is missing, the tests are performed as described by Bretz et al. (2011), i.e. tests of intersection null hypotheses always exhaust the full alpha level even if the sum of weights is strictly smaller than one. If `test="simple-parametric"` the tests are performed as defined in Equation (3) of Bretz et al. (2011).
`upscale`	Logical. If `upscale=FALSE` then for each intersection of hypotheses (i.e. each subgraph) a weighted test is performed at the possibly reduced level alpha of sum(w)*alpha, where sum(w) is the sum of all node weights in this subset. If `upscale=TRUE` all weights are upscaled, so that sum(w)=1.

Value

A vector or a matrix containing the test results for the hypotheses under consideration. Significant tests are denoted by a 1, non-significant results by a 0.

References

Bretz, F., Maurer, W., Brannath, W. and Posch, M. (2009) A graphical approach to sequentially rejective multiple test procedures. Statistics in Medicine, 28, 586–604

Bretz, F., Maurer, W. and Hommel, G. (2010) Test and power considerations for multiple endpoint analyses using sequentially rejective graphical procedures, to appear in Statistics in Medicine

Examples

```
#### example from Bretz et al. (2010)
weights <- c(1/3, 1/3, 1/3, 0, 0, 0)
graph <- rbind(c(0,       0.5, 0,     0.5, 0,      0),
               c(1/3,     0,   1/3,    0,   1/3,    0),
               c(0,       0.5, 0,     0,   0,      0.5),
               c(0,       1,   0,     0,   0,      0),
               c(0.5,     0,   0.5,   0,   0,      0),
               c(0,       1,   0,     0,   0,      0))
pvals <- c(0.1, 0.008, 0.005, 0.15, 0.04, 0.006)
graphTest(pvals, weights, alpha=0.025, graph)

## observe graphical procedure in detail
graphTest(pvals, weights, alpha=0.025, graph, verbose = TRUE)

## now use many p-values (useful for power simulations)
pvals <- matrix(rbeta(6e4, 1, 30), ncol = 6)
out <- graphTest(pvals, weights, alpha=0.025, graph)
head(out)
# example using multiple graphs (instead of 1)
G1 <- rbind(c(0,0.5,0.5,0,0), c(0,0,1,0,0),
            c(0, 0, 0, 1-0.01, 0.01), c(0, 1, 0, 0, 0),
            c(0, 0, 0, 0, 0))
G2 <- rbind(c(0,0,1,0,0), c(0.5,0,0.5,0,0),
            c(0, 0, 0, 0.01, 1-0.01), c(0, 0, 0, 0, 0),
            c(1, 0, 0, 0, 0))
weights <- rbind(c(1, 0, 0, 0, 0), c(0, 1, 0, 0, 0))
pvals <- c(0.012, 0.025, 0.005, 0.0015, 0.0045)
out <- graphTest(pvals, weights, alpha=c(0.0125, 0.0125), G=list(G1, G2), verbose = TRUE)

## now again with many p-values
pvals <- matrix(rbeta(5e4, 1, 30), ncol = 5)
out <- graphTest(pvals, weights, alpha=c(0.0125, 0.0125), G=list(G1, G2))
head(out)
```

---

hGraph *Create multiplicity graphs using ggplot2*

Description

Plots a multiplicity graph defined by user inputs. The graph can also be used with the gMCP package to evaluate a set of nominal p-values for the tests of the hypotheses in the graph.

Usage

```
hGraph(
  nHypotheses = 4,
  nameHypotheses = paste("H", (1:nHypotheses), sep = ""),
  alphaHypotheses = 0.025/nHypotheses,
  m = matrix(array(1/(nHypotheses - 1), nHypotheses^2), nrow = nHypotheses) -
    diag(1/(nHypotheses - 1), nHypotheses),
  fill = 1,
  palette = grDevices::gray.colors(length(unique(fill)), start = 0.5, end = 0.8),
  labels = LETTERS[1:length(unique(fill))],
  legend.name = " ",
  legend.position = "none",
  halfWid = 0.5,
  halfHgt = 0.5,
  trhw = 0.1,
  trhh = 0.075,
  trprop = 1/3,
  digits = 5,
  trdigits = 2,
  size = 6,
  boxtextsize = 4,
  legend.textsize = size * 2.5,
  arrowsize = 0.02,
  radianStart = if ((nHypotheses)%%2 != 0) {
     pi * (1/2 + 1/nHypotheses)
 } else
    {
     pi * (1 + 2/nHypotheses)/2
 },
  offset = pi/4/nHypotheses,
  xradius = 2,
  yradius = xradius,
  x = NULL,
  y = NULL,
  wchar = "\u03b1"
)
```

Arguments

`nHypotheses`	number of hypotheses in graph
`nameHypotheses`	hypothesis names
`alphaHypotheses`	alpha-levels or weights for ellipses
`m`	square transition matrix of dimension ‘nHypotheses‘
`fill`	grouping variable for hypotheses
`palette`	colors for groups
`labels`	text labels for groups

`legend.name`	text for legend header
`legend.position`	text string or x,y coordinates for legend
`halfWid`	half width of ellipses
`halfHgt`	half height of ellipses
`trhw`	transition box width
`trhh`	transition box height
`trprop`	proportion of transition arrow length where transition box is placed
`digits`	number of digits to show for alphaHypotheses
`trdigits`	digits displayed for transition weights
`size`	text size in ellipses
`boxtextsize`	transition text size
`legend.textsize`	legend text size
`arrowsize`	size of arrowhead for transition arrows
`radianStart`	radians from origin for first ellipse; nodes spaced equally in clockwise order with centers on an ellipse by default
`offset`	rotational offset in radians for transition weight arrows
`xradius`	horizontal ellipse diameter on which ellipses are drawn
`yradius`	vertical ellipse diameter on which ellipses are drawn
`x`	x coordinates for hypothesis ellipses if elliptical arrangement is not wanted
`y`	y coordinates for hypothesis ellipses if elliptical arrangement is not wanted
`wchar`	character for alphaHypotheses in ellipses; defaults to the Unicode escape sequence `\u03b1` (Greek letter alpha). See list of Unicode characters for a more comprehensive character list.

Details

See vignette **Multiplicity graphs formatting using ggplot2** for explanation of formatting.

Value

A ‘ggplot‘ object with a multi-layer multiplicity graph

Examples

```
# Use Cairo PDF device for better Unicode character support
# when checking the package. Needed for R >= 4.4.0.
if (names(dev.cur()) == "pdf") {
  fn <- attr(.Device, "filepath")
  fn0 <- "gMCPLite-Ex.pdf"
  if (!is.null(fn) && fn == fn0) {
    dv <- cairo_pdf(fn0)
    on.exit(dev.off(dv), add = TRUE)
  }
}

# Defaults: note clockwise ordering
hGraph(5)
# Add colors (default is 3 gray shades)
hGraph(3,fill=1:3)
# Colorblind palette
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73",
               "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
hGraph(6,fill=as.factor(1:6),palette=cbPalette)
# Use a hue palette
hGraph(4,fill=factor(1:4),palette=scales::hue_pal(l=75)(4))
# different alpha allocation, hypothesis names and transitions
alphaHypotheses <- c(.005,.007,.013)
nameHypotheses <- c("ORR","PFS","OS")
m <- matrix(c(0,1,0,
              0,0,1,
              1,0,0),nrow=3,byrow=TRUE)
hGraph(3,alphaHypotheses=alphaHypotheses,nameHypotheses=nameHypotheses,m=m)
# Custom position and size of ellipses, change text to multi-line text
# Adjust box width
# add legend in middle of plot
hGraph(3,x=sqrt(0:2),y=c(1,3,1.5),size=6,halfWid=.3,halfHgt=.3, trhw=0.6,
       palette=cbPalette[2:4], fill = c(1, 2, 2),
     legend.position = c(.6,.5), legend.name = "Legend:", labels = c("Group 1", "Group 2"),
       nameHypotheses=c("H1:\n Long name","H2:\n Longer name","H3:\n Longest name"))
```

joinGraphs *Joins two graphMCP objects*

Description

Creates a new graphMCP object by joining two given graphMCP objects.

Usage

```
joinGraphs(graph1, graph2, xOffset = 0, yOffset = 200)
```

Arguments

graph1	A graph of class graphMCP.
graph2	A graph of class graphMCP.
xOffset	A numeric specifying an offset (on the x-axis) for placing the nodes and edge labels of the second graph.
yOffset	A numeric specifying an offset (on the y-axis) for placing the nodes and edge labels of the second graph.

Details

If `graph1` and `graph2` have duplicates in the node names, the nodes of the second graph will be renamed.

If and only if the sum of the weights of graph1 and graph2 exceeds 1, the weights are scaled so that the sum equals 1.

A description attribute of either graph will be discarded.

Value

A graphMCP object that represents a graph that consists of the two given graphs.

Author(s)

Kornelius Rohmeyer `<rohmeyer@small-projects.de>`

See Also

`graphMCP`

Examples

```
g1 <- BonferroniHolm(2)
g2 <- BonferroniHolm(3)

suppressWarnings(joinGraphs(g1, g2))
```

`matrix2graph` *Matrix2Graph and Graph2Matrix*

Description

Creates a graph of class `graphMCP` from a given transition matrix or vice versa.

Usage

```
matrix2graph(m, weights = rep(1/dim(m)[1], dim(m)[1]))

graph2matrix(graph)
```

Arguments

`m`	A transition matrix.
`weights`	A numeric for the initial weights.
`graph`	A graph of class `graphMCP`.

Details

The hypotheses names are the row names or if these are `NULL`, the column names or if these are also `NULL` of type H1, H2, H3, ...

If the diagonal of the matrix is unequal zero, the values are ignored and a warning is given.

Value

A graph of class `graphMCP` with the given transition matrix for matrix2graph. The transition matrix of a `graphMCP` graph for graph2matrix.

Author(s)

Kornelius Rohmeyer `<rohmeyer@small-projects.de>`

Examples

```
# Bonferroni-Holm:
m <- matrix(rep(1/3, 16), nrow=4)
diag(m) <- c(0, 0, 0, 0)
graph <- matrix2graph(m)
print(graph)
graph2matrix(graph)
```

`parametric.test` *Weighted parametric test*

Description

It is assumed that under the global null hypothesis $(\Phi^{-1}(1-p_1), ..., \Phi^{-1}(1-p_m))$ follow a multivariate normal distribution with correlation matrix `correlation` where Φ^{-1} denotes the inverse of the standard normal distribution function.

Usage

```
parametric.test(
  pvalues,
  weights,
  alpha = 0.05,
  adjPValues = TRUE,
  verbose = FALSE,
  correlation,
  ...
)
```

Arguments

`pvalues`	A numeric vector specifying the p-values.
`weights`	A numeric vector of weights.
`alpha`	A numeric specifying the maximal allowed type one error rate. If `adjPValues==TRUE` (default) the parameter `alpha` is not used.
`adjPValues`	Logical scalar. If `TRUE` (the default) an adjusted p-value for the weighted parametric test is returned. Otherwise if `adjPValues==FALSE` a logical value is returned whether the null hypothesis can be rejected.
`verbose`	Logical scalar. If `TRUE` verbose output is generated.
`correlation`	Correlation matrix. For parametric tests the p-values must arise from one-sided tests with multivariate normal distributed test statistics for which the correlation is (partially) known. In that case a weighted parametric closed test is performed (also see `generatePvals`). Unknown values can be set to NA. (See details for more information)
`...`	Further arguments possibly passed by `gMCP` which will be used by other test procedures but not this one.

Details

For example, this is the case if $p_1, ..., p_m$ are the raw p-values from one-sided z-tests for each of the elementary hypotheses where the correlation between z-test statistics is generated by an overlap in the observations (e.g. comparison with a common control, group-sequential analyses etc.). An application of the transformation $\Phi^{-1}(1 - p_i)$ to raw p-values from a two-sided test will not in general lead to a multivariate normal distribution. Partial knowledge of the correlation matrix is supported. The correlation matrix has to be passed as a numeric matrix with elements of the form: $correlation[i, i] = 1$ for diagonal elements, $correlation[i, j] = \rho_{ij}$, where ρ_{ij} is the known value of the correlation between $\Phi^{-1}(1 - p_i)$ and $\Phi^{-1}(1 - p_j)$ or `NA` if the corresponding correlation is unknown. For example correlation[1,2]=0 indicates that the first and second test statistic are uncorrelated, whereas correlation[2,3] = NA means that the true correlation between statistics two and three is unknown and may take values between -1 and 1. The correlation has to be specified for complete blocks (ie.: if cor(i,j), and cor(i,j') for i!=j!=j' are specified then cor(j,j') has to be specified as well) otherwise the corresponding intersection null hypotheses tests are not uniquely defined and an error is returned.

For further details see the given references.

Value

adjusted p-value or decision of rejection

References

Bretz F., Posch M., Glimm E., Klinglmueller F., Maurer W., Rohmeyer K. (2011): Graphical approaches for multiple endpoint problems using weighted Bonferroni, Simes or parametric tests. Biometrical Journal 53 (6), pages 894-913, Wiley. doi:10.1002/bimj.201000239

Examples

```
parametric.test(pvalues=c(0.1,0.2,0.05), weights=c(0.5,0.5,0), correlation = diag(3))
parametric.test(pvalues=c(0.1,0.2,0.05), weights=c(0.5,0.5,0), correlation = diag(3),
adjPValues = FALSE)
```

---

permutations *Permutation for a design matrix*

Description

Permutation for a design matrix

Usage

```
permutations(n)
```

Arguments

n	dimension of the matrix

Value

a n*(2^n) dimensional matrix

Examples

```
permutations(3)
```

---

placeNodes *Placement of graph nodes*

Description

Places the nodes of a graph according to a specified layout.

Usage

```
placeNodes(graph, nrow, ncol, byrow = TRUE, topdown = TRUE, force = FALSE)
```

Arguments

graph	A graph of class graphMCP or class entangledMCP.
nrow	The desired number of rows.
ncol	The desired number of columns.
byrow	Logical whether the graph is filled by rows (otherwise by columns).
topdown	Logical whether the rows are filled top-down or bottom-up.
force	Logical whether a graph that has already a layout should be given the specified new layout.

Details

If one of nrow or ncol is not given, an attempt is made to infer it from the number of nodes of the graph and the other parameter. If neither is given, the graph is placed as a circle.

Value

The graph with nodes placed according to the specified layout.

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

See Also

graphMCP, entangledMCP

Examples

```

g <- matrix2graph(matrix(0, nrow=6, ncol=6))

g <- placeNodes(g, nrow=2, force=TRUE)
```

---

rejectNode *Rejects a node/hypothesis and updates the graph accordingly.*

Description

Rejects a node/hypothesis and updates the graph accordingly.

Usage

```
rejectNode(graph, node, upscale = FALSE, verbose = FALSE, keepWeights = FALSE)
```

Arguments

`graph`	A graph of class `graphMCP` or `entangledMCP`.
`node`	A character string specifying the node to reject.
`upscale`	Logical. If `upscale=TRUE` then the weights of all non-rejected nodes are scaled so that the sum is equal to 1. This forces `keepWeights=FALSE` to reduce confusion, since otherwise the sum of weights could become bigger than 1.
`verbose`	Logical scalar. If `TRUE` verbose output is generated during sequentially rejection steps.
`keepWeights`	Logical scalar. If `FALSE` the weight of a node without outgoing edges is set to 0 if it is removed. Otherwise it keeps its weight.

Details

For details see the given references.

Value

An updated graph of class `graphMCP` or `entangledMCP`.

Author(s)

Kornelius Rohmeyer `<rohmeyer@small-projects.de>`

References

Frank Bretz, Willi Maurer, Werner Brannath, Martin Posch: A graphical approach to sequentially rejective multiple test procedures. Statistics in Medicine 2009 vol. 28 issue 4 page 586-604. `https://www.meduniwien.ac.at/fwf_adaptive/papers/bretz_2009_22.pdf`

See Also

`graphMCP`

Examples

```
m <- matrix(0, nrow = 4, ncol = 4)
m[1,3] <- m[2,4] <- m[3,2] <- m[4,1] <- 1
p1 <- c(0.01, 0.005, 0.01, 0.5)
a <- 0.05
w <- c(1/2, 1/2, 0, 0)
g <- matrix2graph(m, w)
gMCP(g, pvalues=p1, alpha=a)
rejectNode(graph = g, node = 4)
```

replaceVariables *Replaces variables in a general graph with specified numeric values*

Description

Given a list of variables and real values a general graph is processed and each variable replaced with the specified numeric value.

Usage

```
replaceVariables(
  graph,
  variables = list(),
  ask = TRUE,
  partial = FALSE,
  expand = TRUE,
  list = FALSE
)
```

Arguments

`graph`	A graph of class `graphMCP` or class `entangledMCP`.
`variables`	A named list with one or more specified real values, for example `list(a=0.5, b=0.8, "tau"=0.5)` or `list(a=c(0.5, 0.8), b=0.8, "tau"=0.5)`. If `ask=TRUE` and this list is missing at all or single variables are missing from the list, the user is asked for the values (if the session is not interactive an error is thrown). For interactively entered values only single numbers are supported.
`ask`	If `FALSE` all variables that are not specified are not replaced.
`partial`	IF `TRUE` only specified variables are replaced and parameter `ask` is ignored.
`expand`	Used internally. Don't use yourself.
`list`	If `TRUE` the result will always be a list, even if only one graph is returned in this list.

Value

A graph or a matrix with variables replaced by the specified numeric values. Or a list of theses graphs and matrices if a variable had more than one value.

Author(s)

Kornelius Rohmeyer `<rohmeyer@small-projects.de>`

See Also

`graphMCP`, `entangledMCP`

Examples

```
graph <- HungEtWang2010()
replaceVariables(graph, list("tau"=0.5,"omega"=0.5, "nu"=0.5))
replaceVariables(graph, list("tau"=c(0.1, 0.5, 0.9),"omega"=c(0.2, 0.8), "nu"=0.4))
```

`simConfint` *Simultaneous confidence intervals for sequentially rejective multiple test procedures*

Description

Calculates simultaneous confidence intervals for sequentially rejective multiple test procedures.

Usage

```
simConfint(
  object,
  pvalues,
  confint,
  alternative = c("less", "greater"),
  estimates,
  df,
  alpha = 0.05,
  mu = 0
)
```

Arguments

`object`	A graph of class `graphMCP`.
`pvalues`	A numeric vector specifying the p-values for the sequentially rejective MTP.
`confint`	One of the following: A character string `"normal"`, `"t"` or a function that calculates the confidence intervals. If `confint=="t"` the parameter `df` must be specified. If `confint` is a function it must be of signature `("character","numeric")`, where the first parameter is the hypothesis name and the second the marginal confidence level (see examples).
`alternative`	A character string specifying the alternative hypothesis, must be "greater" or "less".
`estimates`	Point estimates for the parameters of interest.
`df`	Degree of freedom as numeric.
`alpha`	The overall alpha level as numeric scalar.
`mu`	The numerical parameter vector under null hypothesis.

Details

For details see the given references.

Value

A matrix with columns giving lower confidence limits, point estimates and upper confidence limits for each parameter. These will be labeled as "lower bound", "estimate" and "upper bound". (1-level)/2 in % (by default 2.5% and 97.5%).

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

References

Frank Bretz, Willi Maurer, Werner Brannath, Martin Posch: A graphical approach to sequentially rejective multiple test procedures. Statistics in Medicine 2009 vol. 28 issue 4 page 586-604. https://www.meduniwien.ac.at/fwf_adaptive/papers/bretz_2009_22.pdf

See Also

graphMCP

Examples

```
est <- c("H1"=0.860382, "H2"=0.9161474, "H3"=0.9732953)
# Sample standard deviations:
ssd <- c("H1"=0.8759528, "H2"=1.291310, "H3"=0.8570892)

pval <- c(0.01260, 0.05154, 0.02124)/2

simConfint(BonferroniHolm(3), pvalues=pval,
confint=function(node, alpha) {
c(est[node]-qt(1-alpha,df=9)*ssd[node]/sqrt(10), Inf)
}, estimates=est, alpha=0.025, mu=0, alternative="greater")

# Note that the sample standard deviations in the following call
# will be calculated from the pvalues and estimates.
ci <- simConfint(BonferroniHolm(3), pvalues=pval,
confint="t", df=9, estimates=est, alpha=0.025, alternative="greater")
ci

# plotSimCI(ci)
```

---

simes.on.subsets.test *Simes on subsets, otherwise Bonferroni*

Description

Weighted Simes test introduced by Benjamini and Hochberg (1997)

Usage

```
simes.on.subsets.test(
  pvalues,
  weights,
  alpha = 0.05,
  adjPValues = TRUE,
  verbose = FALSE,
  subsets,
  subset,
  ...
)
```

Arguments

`pvalues`	A numeric vector specifying the p-values.
`weights`	A numeric vector of weights.
`alpha`	A numeric specifying the maximal allowed type one error rate. If `adjPValues==TRUE` (default) the parameter `alpha` is not used.
`adjPValues`	Logical scalar. If `TRUE` (the default) an adjusted p-value for the weighted test is returned. Otherwise if `adjPValues==FALSE` a logical value is returned whether the null hypothesis can be rejected.
`verbose`	Logical scalar. If `TRUE` verbose output is generated.
`subsets`	A list of subsets given by numeric vectors containing the indices of the elementary hypotheses for which the weighted Simes test is applicable.
`subset`	A numeric vector containing the numbers of the indices of the currently tested elementary hypotheses.
`...`	Further arguments possibly passed by `gMCP` which will be used by other test procedures but not this one.

Details

As an additional argument a list of subsets must be provided, that states in which cases a Simes test is applicable (i.e. if all hypotheses to test belong to one of these subsets), e.g. subsets <- list(c("H1", "H2", "H3"), c("H4", "H5", "H6")) Trimmed Simes test for intersections of two hypotheses and otherwise weighted Bonferroni-test

Value

adjusted p-value or decision of rejection

Examples

```
simes.on.subsets.test(pvalues=c(0.1,0.2,0.05), weights=c(0.5,0.5,0))
simes.on.subsets.test(pvalues=c(0.1,0.2,0.05), weights=c(0.5,0.5,0), adjPValues=FALSE)

graph <- BonferroniHolm(4)
pvalues <- c(0.01, 0.05, 0.03, 0.02)
```

```
gMCP.extended(graph=graph, pvalues=pvalues, test=simes.on.subsets.test, subsets=list(1:2, 3:4))
```

`simes.test` *Weighted Simes test*

Description

Weighted Simes test introduced by Benjamini and Hochberg (1997)

Usage

```
simes.test(
  pvalues,
  weights,
  alpha = 0.05,
  adjPValues = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`pvalues`	A numeric vector specifying the p-values.
`weights`	A numeric vector of weights.
`alpha`	A numeric specifying the maximal allowed type one error rate. If `adjPValues==TRUE` (default) the parameter `alpha` is not used.
`adjPValues`	Logical scalar. If `TRUE` (the default) an adjusted p-value for the weighted Simes test is returned. Otherwise if `adjPValues==FALSE` a logical value is returned whether the null hypothesis can be rejected.
`verbose`	Logical scalar. If `TRUE` verbose output is generated.
`...`	Further arguments possibly passed by `gMCP` which will be used by other test procedures but not this one.

Value

adjusted p-value or decision of rejection

Examples

```
simes.test(pvalues=c(0.1,0.2,0.05), weights=c(0.5,0.5,0))
simes.test(pvalues=c(0.1,0.2,0.05), weights=c(0.5,0.5,0), adjPValues=FALSE)
```

subgraph — *Get a subgraph*

Description

Given a set of nodes and a graph this function creates the subgraph containing only the specified nodes.

Usage

```
subgraph(graph, subset)
```

Arguments

`graph`	A graph of class `graphMCP`.
`subset`	A logical or character vector specifying the nodes in the subgraph.

Value

A subgraph containing only the specified nodes.

Author(s)

Kornelius Rohmeyer <`rohmeyer@small-projects.de`>

See Also

`graphMCP`

Examples

```
graph <- improvedParallelGatekeeping()
subgraph(graph, c(TRUE, FALSE, TRUE, FALSE))
subgraph(graph, c("H1", "H3"))
```

`substituteEps` *Substitute Epsilon*

Description

Substitute Epsilon with a given value.

Usage

```
substituteEps(graph, eps = 10^(-3))
```

Arguments

`graph`	A graph of class `graphMCP` or class `entangledMCP`.
`eps`	A numeric scalar specifying a value for epsilon edges.

Details

For details see the given references.

Value

A graph where all epsilons have been replaced with the given value.

Author(s)

Kornelius Rohmeyer `<rohmeyer@small-projects.de>`

See Also

`graphMCP`, `entangledMCP`

Examples

```
graph <- improvedParallelGatekeeping()
graph
substituteEps(graph, eps=0.01)
```

weighted.test.functions *Weighted Test Functions for use with gMCP*

Description

The package gMCP provides the following weighted test functions:

bonferroni.test Bonferroni test - see `?bonferroni.test` for details.

parametric.test Parametric test - see `?parametric.test` for details.

simes.test Simes test - see `?simes.test` for details.

bonferroni.trimmed.simes.test Trimmed Simes test for intersections of two hypotheses and otherwise Bonferroni - see `?bonferroni.trimmed.simes.test` for details.

simes.on.subsets.test Simes test for intersections of hypotheses from certain sets and otherwise Bonferroni - see `?simes.on.subsets.test` for details.

Details

Depending on whether `adjPValues==TRUE` these test functions return different values:

- If `adjPValues==TRUE` the minimal value for alpha is returned for which the null hypothesis can be rejected. If that's not possible (for example in case of the trimmed Simes test adjusted p-values can not be calculated), the test function may throw an error.
- If `adjPValues==FALSE` a logical value is returned whether the null hypothesis can be rejected.

To provide your own test function write a function that takes at least the following arguments:

pvalues A numeric vector specifying the p-values.

weights A numeric vector of weights.

alpha A numeric specifying the maximal allowed type one error rate. If `adjPValues==TRUE` (default) the parameter `alpha` should not be used.

adjPValues Logical scalar. If `TRUE` an adjusted p-value for the weighted test is returned (if possible - if not the function should call `stop`). Otherwise if `adjPValues==FALSE` a logical value is returned whether the null hypothesis can be rejected.

... Further arguments possibly passed by `gMCP` which will be used by other test procedures but not this one.

Further the following parameters have a predefined meaning:

verbose Logical scalar. If `TRUE` verbose output should be generated and printed to the standard output

subset

correlation

Author(s)

Kornelius Rohmeyer <rohmeyer@small-projects.de>

Examples

```
# The test function 'bonferroni.test' is used in by gMCP in the following call:
graph <- BonferroniHolm(4)
pvalues <- c(0.01, 0.05, 0.03, 0.02)
alpha <- 0.05
r <- gMCP.extended(graph=graph, pvalues=pvalues, test=bonferroni.test, verbose=TRUE)

# For the intersection of all four elementary hypotheses this results in a call
bonferroni.test(pvalues=pvalues, weights=getWeights(graph))
bonferroni.test(pvalues=pvalues, weights=getWeights(graph), adjPValues=FALSE)

# bonferroni.test function:
bonferroni.test <- function(pvalues, weights, alpha=0.05, adjPValues=TRUE, verbose=FALSE, ...) {
  if (adjPValues) {
    return(min(pvalues/weights))
  } else {
    return(any(pvalues<=alpha*weights))
  }
}
```

Index