Package ‘multtest’

April 24, 2025

Title Resampling-based multiple hypothesis testing

Version 2.64.0

Author Katherine S. Pollard, Houston N. Gilbert, Yongchao Ge, Sandra Taylor, Sandrine Dudoit

Description Non-parametric bootstrap and permutation resampling-based multiple testing procedures (including empirical Bayes methods) for controlling the family-wise error rate (FWER), generalized family-wise error rate (gFWER), tail probability of the proportion of false positives (TPPFP), and false discovery rate (FDR). Several choices of bootstrap-based null distribution are implemented (centered, centered and scaled, quantile-transformed). Single-step and step-wise methods are available. Tests based on a variety of t- and F-statistics (including t-statistics based on regression parameters from linear and survival models as well as those based on correlation parameters) are included. When probing hypotheses with t-statistics, users may also select a potentially faster null distribution which is multivariate normal with mean zero and variance covariance matrix derived from the vector influence function. Results are reported in terms of adjusted p-values, confidence regions and test statistic cutoffs. The procedures are directly applicable to identifying differentially expressed genes in DNA microarray experiments.

Maintainer Katherine S. Pollard <katherine.pollard@gladstone.ucsf.edu>

Depends R (>= 2.10), methods, BiocGenerics, Biobase

Imports survival, MASS, stats4

Suggests snow

License LGPL

LazyLoad yes

biocViews Microarray, DifferentialExpression, MultipleComparison

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

git_branch RELEASE_3_21

git_last_commit 3b70fb9

git_last_commit_date 2025-04-15

Repository Bioconductor 3.21

Date/Publication 2025-04-23

Contents

`boot.null` *Non-parametric bootstrap resampling function in package 'multtest'*

Description

Given a data set and a closure, which consists of a function for computing the test statistic and its enclosing environment, this function produces a non-parametric bootstrap estimated test statistics null distribution. The observations in the data are resampled using the ordinary non-parametric bootstrap is used to produce an estimated test statistics distribution. This distribution is then transformed to produce the null distribution. Options for transforming the nonparametric bootstrap distribution include `center.only`, `center.scale`, and `quant.trans`. Details are given below. These functions are called by `MTP` and `EBMTP`.

Usage

```
boot.null(X, label, stat.closure, W = NULL, B = 1000, test, nulldist, theta0 = 0, tau0 = 1, marg.null = NU
    ncp = 0, perm.mat, alternative = "two.sided", seed = NULL,
    cluster = 1, dispatch = 0.05, keep.nulldist, keep.rawdist)

boot.resample(X, label, p, n, stat.closure, W, B, test)

center.only(muboot, theta0, alternative)

center.scale(muboot, theta0, tau0, alternative)

quant.trans(muboot, marg.null, marg.par, ncp, alternative, perm.mat)
```

Arguments

- `X` A matrix, data.frame or ExpressionSet containing the raw data. In the case of an ExpressionSet, `exprs(X)` is the data of interest and `pData(X)` may contain outcomes and covariates of interest. For `boot.resample` `X` must be a matrix. For currently implemented tests, one hypothesis is tested for each row of the data.
- `label` A vector containing the class labels for t- and F-tests.
- `stat.closure` A closure for test statistic computation, like those produced internally by the `MTP` function. The closure consists of a function for computing the test statistic and its enclosing environment, with bindings for relevant additional arguments (such as null values, outcomes, and covariates).
- `W` A vector or matrix containing non-negative weights to be used in computing the test statistics. If a matrix, `W` must be the same dimension as `X` with one weight for each value in `X`. If a vector, `W` may contain one weight for each observation (i.e. column) of `X` or one weight for each variable (i.e. row) of `X`. In either case, the weights are duplicated appropriately. Weighted F-tests are not available. Default is 'NULL'.
- `B` The number of bootstrap iterations (i.e. how many resampled data sets) or the number of permutations (if `nulldist` is 'perm'). Can be reduced to increase the speed of computation, at a cost to precision. Default is 1000.
- `test` Character string specifying the test statistics to use. See `MTP` for a list of tests.
- `theta0` The value used to center the test statistics. For tests based on a form of t-statistics, this should be zero (default). For F-tests, this should be 1.
- `tau0` The value used to scale the test statistics. For tests based on a form of t-statistics, this should be 1 (default). For F-tests, this should be 2/(K-1), where K is the number of groups. This argument is missing when `center.only` is chosen for transforming the raw bootstrap test statistics.
- `marg.null` If `nulldist='boot.qt'`, the marginal null distribution to use for quantile transformation. Can be one of 'normal', 't', 'f' or 'perm'. Default is 'NULL', in which case the marginal null distribution is selected based on choice of test statistics. Defaults explained below. If 'perm', the user must supply a vector

or matrix of test statistics corresponding to another marginal null distribution, perhaps one created externally by the user, and possibly referring to empirically derived *marginal permutation distributions*, although the statistics could represent any suitable choice of marginal null distribution.

`marg.par`
: If `nulldist='boot.qt'`, the parameters defining the marginal null distribution in `marg.null` to be used for quantile transformation. Default is 'NULL', in which case the values are selected based on choice of test statistics and other available parameters (e.g., sample size, number of groups, etc.). Defaults explained below. User can override defaults, in which case a matrix of marginal null distribution parameters can be accepted. Providing a matrix of values allows the user to perform multiple testing using parameters which may vary with each hypothesis, as may be desired in common-quantile minP procedures. In this way, factors affecting multiple testing procedure performance such as sample size or missingness may be assessed.

`ncp`
: If `nulldist='boot.qt'`, a value for a possible noncentrality parameter to be used during marginal quantile transformation. Default is 'NULL'.

`perm.mat`
: If `nulldist='boot.qt'` and `marg.null='perm'`, a matrix of user-supplied test statistics from a particular distribution to be used during marginal quantile transformation. The statistics may represent empirically derived marginal permutation values, may be theoretical values, or may represent a sample from some other suitable choice of marginal null distribution.

`alternative`
: Character string indicating the alternative hypotheses, by default 'two.sided'. For one-sided tests, use 'less' or 'greater' for null hypotheses of 'greater than or equal' (i.e. alternative is 'less') and 'less than or equal', respectively.

`seed`
: Integer or vector of integers to be used as argument to `set.seed` to set the seed for the random number generator for bootstrap resampling. This argument can be used to repeat exactly a test performed with a given seed. If the seed is specified via this argument, the same seed will be returned in the seed slot of the MTP object created. Else a random seed(s) will be generated, used and returned. Vector of integers used to specify seeds for each node in a cluster used to to generate a bootstrap null distribution.

`cluster`
: Integer of 1 or a cluster object created through the package snow. With cluster=1, bootstrap is implemented on single node. Supplying a cluster object results in the bootstrap being implemented in parallel on the provided nodes. This option is only available for the bootstrap procedure.

`csnull`
: DEPRECATED as of `multtest` v. 2.0.0 given expanded null distribution options. Previously, this argument was an indicator of whether the bootstrap estimated test statistics distribution should be centered and scaled (to produce a null distribution) or not. If `csnull=FALSE`, the (raw) non-null bootstrap estimated test statistics distribution was returned. If the non-null bootstrap distribution should be returned, this object is now stored in the 'rawdist' slot when `keep.rawdist=TRUE`.

`dispatch`
: The number or percentage of bootstrap iterations to dispatch at a time to each node of the cluster if a computer cluster is used. If dispatch is a percentage, `B*dispatch` must be an integer. If dispatch is an integer, then `B/dispatch` must be an integer. Default is 5 percent.

p	An integer of the number of variables of interest to be tested.
n	An integer of the total number of samples.
muboot	A matrix of bootstrapped test statistics.
keep.nulldist	Logical indicating whether to return the computed bootstrap null distribution, by default 'TRUE'. Not available for nulldist='perm'. Note that this matrix can be quite large.
keep.rawdist	Logical indicating whether to return the computed non-null (raw) bootstrap distribution, by default 'FALSE'. Not available for when using nulldist='perm' or 'ic'. Note that this matrix can become quite large. If one wishes to use subsequent calls to update in which one updates choice of bootstrap null distribution, keep.rawdist must be TRUE. To save on memory, update only requires that one of keep.nulldist or keep.rawdist be 'TRUE'.

Value

A list with the following elements:

rawboot	If keep.rawdist=TRUE, the matrix of non-null, non-transformed bootstrap test statistics. If 'FALSE', an empty matrix with dimension 0-by-0.
muboot	If keep.rawdist=TRUE (default), the matrix of appropriately transformed null test statistics as given by one of center.scale, center.only, or quant.trans. This is the estimated joint test statistics null distribution.

Both list elements rawboot and muboot contain matrices of dimension the number of hypotheses (typically nrow(X)) by the number of bootstrap iterations (B). Each row of muboot is the bootstrap estimated marginal null distribution for a single hypothesis. For boot.null and center.scale, each column of muboot is a centered and scaled resampled vector of test statistics. For boot.null and center.only, each column of muboot is a centered, resampled vector of test statistics.

For boot.null and quant.trans, each column of muboot is a marginal null quantile-transformed resampled vector of test statistics. For each choice of marginal null distribution (defined by marg.null and marg.par), a random sample of size B is drawn and then rearranged based on the ranks of the marginal test statistics bootstrap distribution corresponding to each hypothesis (typically within rows of X). This means that using quant.trans will set the RNG seed ahead by B * the number of hypotheses (similarly, typically nrow(X)). Tie breaks in the marginal non-null bootstrap distribution are implemented inside the internal function marg.samp called by quant.trans. Default values of marg.null and marg.par are available based on choice of test statistics, sample size 'n', and various other parameters. By the time boot.null is called in either the MTP or EBMTP functions, the default marginal null distribution settings have already been formatted and passed in their correct form to boot.null. These default values correspond to:

t.onesamp: t-distribution with df=n-1;

t.twosamp.equalvar: t-distribution with df=n-2;

t.twosamp.unequalvar: N(0,1);

t.pair: t-distribution with df=n-1, where n is the number of unique samples, i.e., the number of observed differences/paired samples;

f: F-distribution with df1=k-1, df2=n-k, for k groups;

f.block: NA. Only available with permutation distribution;

f.twoway: F-distribution with df1=k-1,df2=n-k*l, for k groups and l blocks;

lm.XvsZ: N(0,1);

lm.YvsXZ: N(0,1);

coxph.YvsXZ: N(0,1);

t.cor t-distribution with df=n-2;

z.cor N(0,1).

The above defaults, however, can be overridden by manually setting values of `marg.null` and `marg.par`.

The `rawboot` and `muboot` objects are returned in the slots `rawdist` and `nulldist` of an object of class `MTP` or `EBMTP` when the arguments `keep.rawdist` or `keep.nulldist` to the `MTP` function are TRUE. For `boot.resample` a matrix of bootstrap samples prior to null transformation is returned.

Note

Thank you to Duncan Temple Lang and Peter Dimitrov for suggestions about the code.

Author(s)

Katherine S. Pollard, Houston N. Gilbert, and Sandra Taylor, with design contributions from Sandrine Dudoit and Mark J. van der Laan.

References

M.J. van der Laan, S. Dudoit, K.S. Pollard (2004), Augmentation Procedures for Control of the Generalized Family-Wise Error Rate and Tail Probabilities for the Proportion of False Positives, Statistical Applications in Genetics and Molecular Biology, 3(1). http://www.bepress.com/sagmb/vol3/iss1/art15/

M.J. van der Laan, S. Dudoit, K.S. Pollard (2004), Multiple Testing. Part II. Step-Down Procedures for Control of the Family-Wise Error Rate, Statistical Applications in Genetics and Molecular Biology, 3(1). http://www.bepress.com/sagmb/vol3/iss1/art14/

S. Dudoit, M.J. van der Laan, K.S. Pollard (2004), Multiple Testing. Part I. Single-Step Procedures for Control of General Type I Error Rates, Statistical Applications in Genetics and Molecular Biology, 3(1). http://www.bepress.com/sagmb/vol3/iss1/art13/

Katherine S. Pollard and Mark J. van der Laan, "Resampling-based Multiple Testing: Asymptotic Control of Type I Error and Applications to Gene Expression Data" (June 24, 2003). U.C. Berkeley Division of Biostatistics Working Paper Series. Working Paper 121. http://www.bepress.com/ucbbiostat/paper121

M.J. van der Laan and A.E. Hubbard (2006), Quantile-function Based Null Distributions in Resampling Based Multiple Testing, Statistical Applications in Genetics and Molecular Biology, 5(1). http://www.bepress.com/sagmb/vol5/iss1/art14/

S. Dudoit and M.J. van der Laan. Multiple Testing Procedures and Applications to Genomics. Springer Series in Statistics. Springer, New York, 2008.

See Also

corr.null, MTP, MTP-class, EBMTP, EBMTP-class, get.Tn, ss.maxT, mt.sample.teststat,get.Tn, wapply, boot.resample

Examples

```
set.seed(99)
data<-matrix(rnorm(90),nr=9)

#closure
ttest<-meanX(psi0=0,na.rm=TRUE,standardize=TRUE,alternative="two.sided",robust=FALSE)

#test statistics
obs<-get.Tn(X=data,stat.closure=ttest,W=NULL)

#bootstrap null distribution (B=100 for speed, default nulldist, "boot.cs")
nulldistn<-boot.null(X=data,W=NULL,stat.closure=ttest,B=100,test="t.onesamp",
 nulldist="boot.cs",theta0=0,tau0=1,alternative="two.sided",
 keep.nulldist=TRUE,keep.rawdist=FALSE)$muboot

#bootstrap null distribution with marginal quantile transformation showing
#default values that are passed to marg.null and marg.par arguments
nulldistn.qt<-boot.null(X=data,W=NULL,stat.closure=ttest,B=100,test="t.onesamp",
 nulldist="boot.qt",theta0=0,tau0=1,alternative="two.sided",
 keep.nulldist=TRUE,keep.rawdist=FALSE,marg.null="t",
 marg.par=matrix(9,nr=10,nc=1))$muboot

#unadjusted p-values
rawp<-apply((obs[1,]/obs[2,])<=nulldistn,1,mean)
sum(rawp<=0.01)

rawp.qt<-apply((obs[1,]/obs[2,])<=nulldistn.qt,1,mean)
sum(rawp.qt<=0.01)
```

---

corr.null — *Function to estimate a test statistics joint null distribution for t-statistics via the vector influence curve*

Description

For a broad class of testing problems, such as the test of single-parameter null hypotheses using t-statistics, a proper, asymptotically valid test statistics joint null distribution is the multivariate Gaussian distribution with mean vector zero and covariance matrix equal to the correlation matrix of the vector influence curve for the estimator of the parameter of interest. The function `corr.null` estimates the correlation matrix of the vector influence curve for such parameters and returns samples from the corresponding normal distribution. Arguments to the function allow for refinements in calculating the resulting null distribution estimate.

Usage

```
corr.null(X, W = NULL, Y = NULL, Z = NULL, test = "t.twosamp.unequalvar",
   alternative = "two-sided", use = "pairwise", B = 1000, MVN.method = "mvrnorm",
    penalty = 1e-06, ic.quant.trans = FALSE, marg.null = NULL,
    marg.par = NULL, perm.mat = NULL)
```

Arguments

- `X` — A matrix, data.frame or ExpressionSet containing the raw data. In the case of an ExpressionSet, `exprs(X)` is the data of interest and `pData(X)` may contain outcomes and covariates of interest. For most currently implemented tests (exception: tests involving correlation parameters), one hypothesis is tested for each row of the data.
- `W` — A matrix containing non-negative weights to be used in computing the test statistics. Must be same dimension as `X`.
- `Y` — A vector, factor, or `Surv` object containing the outcome of interest.
- `Z` — A vector, factor, or matrix containing covariate data to be used in linear regression models. Each variable should be in one column, so that `nrow(Z)=ncol(X)`. By the time the function is called, this argument contains a 'design matrix' with the variable to be tested in the first column, additional covariates in the remaining columns, and no intercept column.
- `test` — Character string specifying the test statistics to use, by default 't.twosamp.unequalvar'. See details (below) for a list of tests.
- `alternative` — Character string indicating the alternative hypotheses, by default 'two.sided'. For one-sided tests, use 'less' or 'greater' for null hypotheses of 'greater than or equal' (i.e. alternative is 'less') and 'less than or equal', respectively.
- `use` — Similar to the options in `cor`, a character string giving a method for computing covariances in the presence of missing values. Default is 'pairwise', which allows for the covariance/correlation matrix to be calculated using the most information possible when `NA`s are present.
- `B` — The number of samples to be drawn from the normal distribution. Default is 1000.
- `MVN.method` — Character string of either of 'mvrnorm' or 'Cholesky' designating how correlated normal test statistics are to be generated. Selecting 'mvrnorm' uses the function of the same name found in the `MASS` library, whereas 'Cholesky' relies on a Cholesky decomposition. Default is 'mvrnorm'.
- `penalty` — If `MVN.method='Cholesky'`, the value in `penalty` is added to all diagonal elements of the estimated test statistics correlation matrix to ensure that the matrix is positive definite and that internal calls to `'chol'` do not return an error. Default is 1e-6.
- `ic.quant.trans` — A logical indicating whether or not a marginal quantile transformation using a t-distribution or user-supplied marginal distribution (stored in `perm.mat`) should be applied to the multivariate normal null distribution. Defaults for `marg.null` and `marg.par` exist, but can also be specified by the user (see below). Default is 'FALSE'.

`marg.null` If `ic.quant.trans=TRUE`, a character string naming the marginal null distribution to use for quantile transformation. Can be one of, 't' or 'perm'. Default is 'NULL', in which case the marginal null distribution is selected based on choice of test statistics. Defaults explained below. If 'perm', the user must supply a vector or matrix of test statistics corresponding to another marginal null distribution, perhaps one created externally by the user, and possibly referring to empirically derived *marginal permutation distributions*, although the statistics could represent any suitable choice of marginal null distribution.

`marg.par` If `ic.quant.trans=TRUE`, the parameters defining the marginal null distribution in `marg.null` to be used for quantile transformation. Default is 'NULL', in which case the values are selected based on choice of test statistics and other available parameters (e.g., sample size, number of groups, etc.). Defaults explained below. User can override defaults, in which case a matrix of marginal null distribution parameters must be provided. Providing a matrix allows the user to perform multiple testing using parameters which may vary with each hypothesis, as may be desired in common-quantile minP procedures

`perm.mat` If `ic.quant.trans=TRUE`, a matrix of user-supplied test statistics from a particular distribution to be used during marginal quantile transformation. Supplying a vector of test statistics will apply the same vector to each hypothesis. The statistics may represent empirically derived marginal permutation values, may be theoretical values, or may represent a sample from some other suitable choice of marginal null distribution.

Details

This function is called internally when the argument `nulldist='ic'` is evaluated in the main user-level functions `MTP` or `EBMTP`. Formatting of the data objects `X`, `W`, `Y`, and especially `Z` occurs at execution begin of the main user-level functions.

Based on the value of `test`, the appropriate correlation matrix of the vector influence curve is calculated. Once the correlation matrix is obtained, one may sample vectors of null test statistics directly from a multivariate normal distribution rather than relying on permutation-based or bootstrap-based resampling. Because the Gaussian distribution is continuous, we expect this choice of null distribution to suffer less from discreteness than either the permutation or the bootstrap distribution. Additionally, in large-scale settings, use of null distributions derived from the vector influence function typically reduce computational bottlenecks associated with resampling methods.

Because the influence curve null distributions have been implemented for parametric, standardized t-statistics, the options `robust` and `standardize` are not allowed. Influence curve null distributions are available for the following values of `test`: 't.onesamp', 't.pair', 't.twosamp.equalvar', 't.twosamp.unequalvar', 'lm.XvsZ', 'lm.YvsXZ', 't.cor', and 'z.cor'.

In the simpler cases involving one-sample and two-sample tests of means, the correlation matrices are obtained via calls to `cor`. For two-sample tests, the correlation matrix corresponds to the following transformation of the group-specific covariance matrices: cov(X(group1))/n1 + cov(X(group2))/n2, where n1 and n2 are sample sizes of each group. When weights are present, the internal function `IC.CorXW.NA` is called to calculate weighted estimates of the (group) covariance matrices from each subject's estimated vector influence curve. The calculations are similar in spirit to those in `cov.wt`, but they are done in a way which allows for handling `NA` elements in the estimated vector influence curve IC_n. The correlation matrix corresponding to IC_n * (IC_n)^t is calculated.

For linear regression models, `corr.null` calculates the vector influence curve associated with each subject/sample. The vector has length equal to the number of hypotheses. The internal function `IC.Cor.NA` is used to calculate IC_n * (IC_n)^t in a manner which allows for NA-handling when the influence curve may contain missing elements. For linear regression models of the form E[Y|X], IC_n takes the form (E[((X^t)X)^(-1)] (X^t)_i Y_i) - Y_i-hat. Influence curves for correlation parameters are more complicated, and the user is referred to the references below.

Once the correlation matrix sigma' corresponding to the variance covariance matrix of the vector influence curve sigma =IC_n * (IC_n)^t is obtained, one may sample from N(0,sigma') to obtain null test statistics.

If `ic.quant.trans=TRUE`, the matrix of null test statistics can be quantile transformed to produce a matrix which accounts for the joint dependencies between test statistics (down columns), but which has marginal t-distributions (across rows). If `marg.null` and `marg.par` are not specified (=NULL), the following default t-distributions are applied:

t.onesamp df=n-1;

t.pair df=n-1, where n is the number of unique samples, i.e., the number of observed differences between paired samples;

t.twosamp.equalvar df=n-2;

t.twosamp.unequalvar df=n-1; N.B., this is not recommended, since the effective degrees of freedom are unknown. With sufficiently large n, a normal approximation should yield similar results.

lm.XvsZ df=n-p, where p is the number of variables in the regression equation;

lm.YvsXZ df=n-p, where p is the number of variables in the regression equation;

t.cor df=n-2;

z.cor N.B., also not recommended. Fisher's z-statistics are already normally distributed. Marginal transformation to a t-distribution makes little sense.

Value

A matrix of null test statistics with dimension the number of hypotheses (typically `nrow(X)`) by the number of desired samples (`B`).

Author(s)

Houston N. Gilbert

References

K.S. Pollard and Mark J. van der Laan, "Resampling-based Multiple Testing: Asymptotic Control of Type I Error and Applications to Gene Expression Data" (June 24, 2003). U.C. Berkeley Division of Biostatistics Working Paper Series. Working Paper 121. http://www.bepress.com/ucbbiostat/paper121

S. Dudoit and M.J. van der Laan. Multiple Testing Procedures and Applications to Genomics. Springer Series in Statistics. Springer, New York, 2008.

H.N. Gilbert, M.J. van der Laan, and S. Dudoit, "Joint Multiple Testing Procedures for Inferring Genetic Networks from Lower-Order Conditional Independence Graphs" (2009). *In preparation.*

See Also

`boot.null`,`MTP`, `MTP-class`, `EBMTP`, `EBMTP-class`, `get.Tn`, `ss.maxT`, `mt.sample.teststat`,`get.Tn`, `wapply`, `boot.resample`

Examples

```
set.seed(99)
data <- matrix(rnorm(10*50),nr=10,nc=50)
nulldistn.mvrnorm <- corr.null(data,t="t.onesamp",alternative="greater",B=5000)
nulldistn.chol <- corr.null(data,t="t.onesamp",MVN.method="Cholesky",penalty=1e-9)
nulldistn.t <- corr.null(data,t="t.onesamp",ic.quant.trans=TRUE)
dim(nulldistn.mvrnorm)
```

`EBMTP` *A function to perform empirical Bayes resampling-based multiple hypothesis testing*

Description

A user-level function to perform empirical Bayes multiple testing procedures (EBMTP). A variety of t- and F-tests, including robust versions of most tests, are implemented. A common-cutoff method is used to control the chosen type I error rate (FWER, gFWER, TPPFP, or FDR). Bootstrap-based null distributions are available. Additionally, for t-statistics, one may wish to sample from an appropriate multivariate normal distribution with mean zero and correlation matrix derived from the vector influence function. In EBMTP, realizations of local q-values, obtained via density estimation, are used to partition null and observed test statistics into guessed sets of true and false null hypotheses at each round of (re)sampling. In this manner, parameters of any type I error rate which can be expressed as a function the number of false positives and true positives can be estimated. Arguments are provided for user control of output. Gene selection in microarray experiments is one application.

Usage

```
EBMTP(X, W = NULL, Y = NULL, Z = NULL, Z.incl = NULL, Z.test = NULL,
    na.rm = TRUE, test = "t.twosamp.unequalvar", robust = FALSE,
    standardize = TRUE, alternative = "two.sided", typeone = "fwer",
    method = "common.cutoff", k = 0, q = 0.1, alpha = 0.05, smooth.null = FALSE,
    nulldist = "boot.cs", B = 1000, psi0 = 0, marg.null = NULL,
    marg.par = NULL, ncp = NULL, perm.mat = NULL, ic.quant.trans = FALSE,
    MVN.method = "mvrnorm", penalty = 1e-06, prior = "conservative",
    bw = "nrd", kernel = "gaussian", seed = NULL, cluster = 1,
    type = NULL, dispatch = NULL, keep.nulldist = TRUE, keep.rawdist = FALSE,
    keep.falsepos = FALSE, keep.truepos = FALSE, keep.errormat = FALSE,
   keep.Hsets=FALSE, keep.margpar = TRUE, keep.index = FALSE, keep.label = FALSE)
```

Arguments

For brevity, the presentation of arguments below will highlight those which differ significantly from arguments in the other main-level user function `MTP`. See `MTP` for further details.

`typeone`	Character string indicating which type I error rate to control, by default family-wise error rate ('fwer'). Other options include generalized family-wise error rate ('gfwer'), with parameter `k` giving the allowed number of false positives, and tail probability of the proportion of false positives ('tppfp'), with parameter `q` giving the allowed proportion of false positives. The false discovery rate ('fdr') can also be controlled. In particular, for 'gfwer', 'tppfp' and 'fdr', multiple testing is not performed via augmentation of the results of a FWER-controlling MTP. Rather, using guessed sets of true and false null hypotheses, these error rates are controlled in a more direct manner.
`method`	Character string indicating the EBMTP method. Currently only 'common.cutoff' is implemented. This method is most similar to 'ss.maxT' in `MTP`.
`nulldist`	Character string indicating which resampling method to use for estimating the joint test statistics null distribution, by default the non-parametric bootstrap with centering and scaling ('boot.cs'). The old default 'boot' will still compile and will correspond to 'boot.cs'. Other null distribution options include 'boot.ctr', 'boot.qt', and 'ic', corresponding to the centered-only bootstrap distribution, quantile-transformed bootstrap distribution, and influence curve multivariate normal joint null distribution, respectively. The permutation distribution is not available.
`prior`	Character string indicating which choice of prior probability to use for estimating local q-values (i.e., the posterior probabilities of a null hypothesis being true given the value of its corresponding test statistic). Default is 'conservative', in which case the prior is set to its most conservative value of 1, meaning that all hypotheses are assumed to belong to the set of true null hypotheses. Other options include 'ABH' for the adaptive Benjamini-Hochberg estimator of the number/proportion of true null hypotheses, and 'EBLQV' for the empirical Bayes local q-value value estimator of the number/proportion of true null hypotheses. If 'EBLQV', the estimator of the prior probability is taken to be the sum of the estimated local q-values divided by the number of tests. Relaxing the

prior may result in more rejections, albeit at a cost of type I error control under certain conditions. See details and references.

bw
: A character string argument to density indicating the smoothing bandwidth to be used during kernel density estimation. Default is 'nrd'.

kernel
: A character string argument to density specifying the smoothing kernel to be used. Default is 'gaussian'.

keep.falsepos
: A logical indicating whether or not to store the matrix of guessed false positives at each round of (re)sampling. The matrix has rows equal to the number of cut-offs (observed test statistics) and columns equal to the B number of bootstrap samples or samples from the multivariate normal distribution (if nulldist='ic'). Default is 'FALSE'.

keep.truepos
: A logical indicating whether or not to store the matrix of guessed true positives at each round of (re)sampling. The matrix has rows equal to the number of cut-offs (observed test statistics) and columns equal to the B number of bootstrap samples or samples from the multivariate normal distribution (if nulldist='ic'). Default is 'FALSE'.

keep.errormat
: A logical indicating whether or not to store the matrix of type I error rate values at each round of (re)sampling. The matrix has rows equal to the number of cut-offs (observed test statistics) and columns equal to the B number of bootstrap samples or samples from the multivariate normal distribution (if nulldist='ic'). Default is 'FALSE'. In the case of FDR-control, for example, this matrix is falsepos/(falsepos + truepos). The row means of this matrix are eventually used for assigning/ordering adjusted p-values to test statistics of each hypothesis.

keep.Hsets
: A logical indicating whether or not to return the matrix of indicators which partition the hypotheses into guessed sets of true and false null hypotheses at each round of (re)sampling. Default is 'FALSE'.

X, W, Y, Z, Z.incl, Z.test, na.rm, test, robust, standardize, alternative, k, q, alpha, smooth.null, B, psi0, marg.null, marg.par, ncp, perm.mat, ic.quant.trans, MVN.method, penalty, seed, cluster, type, dispatch, keep.nulldist, keep.rawdist, keep.margpar, keep.index, keep.label
: These arguments are all similarly used by the MTP function, and their use has been defined elsewhere. Please consult the link{MTP} help file or the references for further details. Note that the MTP-function arguments get.cr, get.cutoff, get.adjp are now DEPRECATED in the EBMTP function. Only adjusted p-values are calculated by EBMTP. These adjusted p-values are returned in the same order as the original hypotheses and raw p-values (typically corresponding to rows of X.)

Details

The EBMTP begins with a marginal nonparametric mixture model for estimating local q-values. By definition, q-values are 'the opposite' of traditional p-values. That is, q-values represent the probability of null hypothesis being true given the value of its corresponding test statistic. If the test statistics Tn have marginal distribution f = pi*f_0 + (1-pi)f_1, where pi is the prior probability of a true null hypothesis and f_0 and f_1 represent the marginal null and alternative densities, respectively, then the local q-value function is given by pi*f_0(Tn)/f(Tn).

One can estimate both the null density f_0 and full density f by applying kernel density estimation over the matrix of null test statistics and the vector of observed test statistics, respectively. Practically, this step in EBMTP also ensures that sidedness is correctly accounted for among the test statistics and their estimated null distribution. The prior probability pi can be set to its most conservative value of 1 or estimated by some other means, e.g., using the adaptive Benjamini Hochberg ('ABH') estimator or by summing up the estimated local q-values themselves ('EBLQV')and dividing by the number of tests. Bounding these estimated probabilities by one provides a vector of estimated local q-values with length equal to the number of hypotheses. Bernoulli 0/1 realizations of the posterior probabilities indicate which hypotheses are guessed as belonging to the true set of null hypotheses given the value of their test statistics. Once this partitioning has been achieved, one can count the numbers of guessed false positives and guessed true positives at each round of (re)sampling that are obtained when using the value of an observed test statistic as a cut-off.

EBMTPs use function closures to represent type I error rates in terms of their defining features. Restricting the choice of type I error rate to 'fwer', 'gfwer', 'tppfp', and 'fdr', means that these features include whether to control the number of false positives or the proportion of false positives among the number of rejetions made (i.e., the false discovery proportion), whether we are controlling a tail probability or expected value error rate, and, in the case of tail probability error rates, what bound we are placing on the random variable defining the type I error rate (e.g., k for 'gfwer' or 'q' for 'tppfp'). Averaging the type I error results over B (bootstrap or multivariate normal) samples provides an estimator of the evidence against the null hypothesis (adjusted p-values) with respect to the choice of type I error rate. Finally, a monotonicity constraint is placed on the adjusted p-values before being returned as output.

As detailed in the references, relaxing the prior may result in a more powerful multiple testing procedure, albeit sometimes at the cost of type I error control. Additionally, when the proportion of true null hypotheses is close to one, type I error control may also become an issue, even when using the most conservative prior probability of one. This feature is known to occur with some other procedures which rely on the marginal nonparametric mixture model for estimating (local) q-values. The slot `EB.h0M` returned by objects of class `EBMTP` is the sum of the local q-values estimated via kernel density estimation (divided by the total number of tests). If this value is close to one (>0.9-0.95), the user will probably not want relax the prior, as even the conservative EBMTP might be approaching a performance bound with respect to type I error control. The user is advised to begin by using the most 'conservative' prior, assess the estimated proportion of true null hypotheses, and then decide if relaxing the prior might be desired. Gains in power over other multiple testing procedures have been observed even when using the most conservative prior of one.

Situations of moderate-high to high levels of correlation may also affect the results of multiple testing methods which use the same mixture model for generating q-values. Microarray analysis represents a scenario in which dependence structures are typically weak enough to mitigate this concern. On the other hand, the analysis of densely sampled SNPs, for example, may present problems.

Value

An object of class `EBMTP`. Again, for brevity, the values below represent slots which distinguish objects of class `EBMTP` from those of class `MTP`.

falsepos — A matrix with rows equal to the number of hypotheses and columns the number of samples of null test statistics (B) indicating the number of guessed false positives when using the corresponding value of the observed test statistic as a cut-off. Not returned unless keep.falsepos=TRUE.

truepos — A matrix with rows equal to the number of hypotheses and columns the number of samples of null test statistics (B) indicating the number of guessed true positives when using the corresponding value of the observed test statistic as a cut-off. Not returned unless keep.truepos=TRUE.

errormat — The matrix obtained after applying to type I error rate function closure to the matrices in falsepos, and, if applicable, truepos. Not returned unless keep.errormat=TRUE.

EB.h0M — The sum of the local q-values obtained after density estimation. This number serves as an estimate of the proportion of true null hypotheses. Values close to one indicate situations in which type I error control may not be guaranteed by the EBMTP. When prior='EBLQV', this value is used as the prior 'pi' during evaluation of the local q-value function.

prior — The numeric value of the prior 'pi' used when evaluating the local q-value function.

prior.type — Character string returning the value of prior in the original call to EBMTP. One of 'conservative', 'ABH', or 'EBLQV'.

lqv — A numeric vector of length the number of hypotheses with the estimated local q-values used for generating guessed sets of true null hypotheses.

Hsets — A numeric matrix with the same dimension as nulldist, containing the Bernoulli realizations of the estimated local q-values stored in lqv which were used to partition the hypotheses into guessed sets of true and false null hypotheses at each round of (re)sampling. Not returned unless keep.Hsets=TRUE.

Author(s)

Houston N. Gilbert, based on the original MTP code written by Katherine S. Pollard

References

H.N. Gilbert, K.S. Pollard, M.J. van der Laan, and S. Dudoit (2009). Resampling-based multiple hypothesis testing with applications to genomics: New developments in R/Bioconductor package multtest. *Journal of Statistical Software* (submitted). Temporary URL: http://www.stat.berkeley.edu/~houston/JSSNullDistEBMTP.pdf.

Y. Benjamini and Y. Hochberg (2000). On the adaptive control of the false discovery rate in multiple testing with independent statistics. *J. Behav. Educ. Statist*. Vol 25: 60-83.

Y. Benjamini, A. M. Krieger and D. Yekutieli (2006). Adaptive linear step-up procedures that control the false discovery rate. *Biometrika*. Vol. 93: 491-507.

M.J. van der Laan, M.D. Birkner, and A.E. Hubbard (2005). Empirical Bayes and Resampling Based Multiple Testing Procedure Controlling the Tail Probability of the Proportion of False Positives. Statistical Applications in Genetics and Molecular Biology, 4(1). http://www.bepress.

com/sagmb/vol4/iss1/art29/

S. Dudoit and M.J. van der Laan. Multiple Testing Procedures and Applications to Genomics. Springer Series in Statistics. Springer, New York, 2008.

S. Dudoit, H.N. Gilbert, and M J. van der Laan (2008). Resampling-based empirical Bayes multiple testing procedures for controlling generalized tail probability and expected value error rates: Focus on the false discovery rate and simulation study. *Biometrical Journal*, 50(5):716-44. http://www.stat.berkeley.edu/~houston/BJMCPSupp/BJMCPSupp.html.

H.N. Gilbert, M.J. van der Laan, and S. Dudoit. Joint multiple testing procedures for graphical model selection with applications to biological networks. Technical report, U.C. Berkeley Division of Biostatistics Working Paper Series, April 2009. URL http://www.bepress.com/ucbbiostat/paper245.

See Also

MTP, EBMTP-class, EBMTP-methods, Hsets

Examples

```
set.seed(99)
data<-matrix(rnorm(90),nr=9)
group<-c(rep(1,5),rep(0,5))

#EB fwer control with centered and scaled bootstrap null distribution
#(B=100 for speed)
eb.m1<-EBMTP(X=data,Y=group,alternative="less",B=100,method="common.cutoff")
print(eb.m1)
summary(eb.m1)
par(mfrow=c(2,2))
plot(eb.m1,top=9)
```

EBMTP-class *Class "EBMTP", classes and methods for empirical Bayes multiple testing procedure output*

Description

An object of class EBMTP is the output of a particular multiple testing procedure, as generated by the function EBMTP. The object has slots for the various data used to make multiple testing decisions, in particular adjusted p-values.

Objects from the Class

Objects can be created by calls of the form
new('MTP',
statistic = ..., object of class numeric
estimate = ..., object of class numeric
sampsize = ..., object of class numeric
rawp = ..., object of class numeric
adjp = ..., object of class numeric
reject = ..., object of class matrix
rawdist = ..., object of class matrix
nulldist = ..., object of class matrix
nulldist.type = ..., object of class character
marg.null = ..., object of class character
marg.par = ..., object of class matrix
label = ..., object of class numeric
falsepos = ..., object of class matrix
truepos = ..., object of class matrix
errormat = ..., object of class matrix
EB.h0M = ..., object of class numeric
prior = ..., object of class numeric
prior.type= ...., object of class character
lqv = ..., object of class numeric
Hsets = ..., object of class matrix
index = ..., object of class matrix
call = ..., object of class call
seed = ..., object of class integer
)

Slots

`statistic` Object of class `numeric`, observed test statistics for each hypothesis, specified by the values of the `MTP` arguments `test`, `robust`, `standardize`, and `psi0`.

`estimate` For the test of single-parameter null hypotheses using t-statistics (i.e., not the F-tests), the numeric vector of estimated parameters corresponding to each hypothesis, e.g. means, differences in means, regression parameters.

`sampsize` Object of class `numeric`, number of columns (i.e. observations) in the input data set.

`rawp` Object of class `numeric`, unadjusted, marginal p-values for each hypothesis.

`adjp` Object of class `numeric`, adjusted (for multiple testing) p-values for each hypothesis (computed only if the `get.adjp` argument is TRUE).

`reject` Object of class `'matrix'`, rejection indicators (TRUE for a rejected null hypothesis), for each value of the nominal Type I error rate `alpha`.

`rawdist` The numeric matrix for the estimated nonparametric non-null test statistics distribution (returned only if `keep.rawdist=TRUE` and if `nulldist` is one of 'boot.ctr', 'boot.cs', or 'boot.qt'). This slot must not be empty if one wishes to call `update` to change choice of bootstrap-based null distribution.

- `nulldist` The numeric matrix for the estimated test statistics null distribution (returned only if `keep.nulldist=TRUE`). By default (i.e., for `nulldist='boot.cs'`), the entries of `nulldist` are the null value shifted and scaled bootstrap test statistics, with one null test statistic value for each hypothesis (rows) and bootstrap iteration (columns).
- `nulldist.type` Character value describing which choice of null distribution was used to generate the MTP results. Takes on one of the values of the original `nulldist` argument in the call to MTP, i.e., 'boot.cs', 'boot.ctr', 'boot.qt', or 'ic'.
- `marg.null` If `nulldist='boot.qt'`, a character value returning which choice of marginal null distribution was used by the MTP. Can be used to check default values or to ensure manual settings were correctly applied.
- `marg.par` If `nulldist='boot.qt'`, a numeric matrix returning the parameters of the marginal null distribution(s) used by the MTP. Can be used to check default values or to ensure manual settings were correctly applied.
- `falsepos` A matrix with rows equal to the number of hypotheses and columns the number of samples of null test statistics (`B`) indicating the number of guessed false positives when using the corresponding value of the observed test statistic as a cut-off. Not returned unless `keep.falsepos=TRUE`.
- `truepos` A matrix with rows equal to the number of hypotheses and columns the number of samples of null test statistics (`B`) indicating the number of guessed true positives when using the corresponding value of the observed test statistic as a cut-off. Not returned unless `keep.truepos=TRUE`.
- `errormat` The matrix obtained after applying to type I error rate function closure to the matrices in `falsepos`, and, if applicable, `truepos`. Not returned unless `keep.errormat=TRUE`.
- `EB.h0M` The sum of the local q-values obtained after density estimation. This number serves as an estimate of the proportion of true null hypotheses. Values close to one indicate situations in which type I error control may not be guaranteed by the EBMTP. When `prior='EBLQV'`, this value is used as the prior 'pi' during evaluation of the local q-value function.
- `prior` The numeric value of the prior 'pi' used when evaluating the local q-value function.
- `prior.type` Character string returning the value of `prior` in the original call to `EBMTP`. One of 'conservative', 'ABH', or 'EBLQV'.
- `lqv` A numeric vector of length the number of hypotheses with the estimated local q-values used for generating guessed sets of true null hypotheses.
- `Hsets` A numeric matrix with the same dimension as `nulldist`, containing the Bernoulli realizations of the estimated local q-values stored in `lqv` which were used to partition the hypotheses into guessed sets of true and false null hypotheses at each round of (re)sampling. Not returned unless `keep.Hsets=TRUE`.
- `label` If `keep.label=TRUE`, a vector storing the values used in the argument `Y`. Storing this object is particularly important when one wishes to update EBMTP objects with F-statistics using default `marg.null` and `marg.par` settings when `nulldist='boot.qt'`.
- `index` For tests of correlation parameters a matrix corresponding to `t(combn(p,2))`, where `p` is the number of variables in `X`. This matrix gives the indices of the variables considered in each pairwise correlation. For all other tests, this slot is empty, as the indices are in the same order as the rows of `X`.
- `call` Object of class `call`, the call to the MTP function.

`seed` An integer or vector for specifying the state of the random number generator used to create the resampled datasets. The seed can be reused for reproducibility in a repeat call to `MTP`. This argument is currently used only for the bootstrap null distribution (i.e., for `nulldist="boot.xx"`). See `?set.seed` for details.

Methods

`signature(x = "EBMTP")`

[: Subsetting method for `EBMTP` class, which operates selectively on each slot of an `EBMTP` instance to retain only the data related to the specified hypotheses.

as.list : Converts an object of class `EBMTP` to an object of class `list`, with an entry for each slot.

plot : plot methods for `EBMTP` class, produces the following graphical summaries of the results of a EBMTP. The type of display may be specified via the `which` argument.

1. Scatterplot of number of rejected hypotheses vs. nominal Type I error rate.

2. Plot of ordered adjusted p-values; can be viewed as a plot of Type I error rate vs. number of rejected hypotheses.

3. Scatterplot of adjusted p-values vs. test statistics (also known as "volcano plot").

4. Plot of unordered adjusted p-values.

The plot method for objects of class `EBMTP` does not return the plots associated with `which=5` (using confidence regions) or with `which=6` (pertaining to cut-offs) as it does for objects of class `MTP`. This is because the function `EBMTP` currently only returns adjusted p-values. The argument `logscale` (by default equal to FALSE) allows one to use the negative decimal logarithms of the adjusted p-values in the second, third, and fourth graphical displays. The arguments `caption` and `sub.caption` allow one to change the titles and subtitles for each of the plots (default subtitle is the MTP function call). Note that some of these plots are implemented in the older function `mt.plot`.

print : print method for `EBMTP` class, returns a description of an object of class `EBMTP`, including sample size, number of tested hypotheses, type of test performed (value of argument `test`), Type I error rate (value of argument `typeone`), nominal level of the test (value of argument `alpha`), name of the EBMTP (value of argument `method`), call to the function `EBMTP`.

In addition, this method produces a table with the class, mode, length, and dimension of each slot of the `EBMTP` instance.

summary : summary method for `EBMTP` class, provides numerical summaries of the results of an EBMTP and returns a list with the following three components.

1. rejections: A data.frame with the number(s) of rejected hypotheses for the nominal Type I error rate(s) specified by the `alpha` argument of the function `MTP`.

2. index: A numeric vector of indices for ordering the hypotheses according to first `adjp`, then `rawp`, and finally the absolute value of `statistic` (not printed in the summary).

3. summaries: When applicable (i.e., when the corresponding quantities are returned by MTP), a table with six number summaries of the distributions of the adjusted p-values, unadjusted p-values, test statistics, and parameter estimates.

EBupdate : update method for EBMTP class, provides a mechanism to re-run the MTP with different choices of the following arguments - nulldist, alternative, typeone, k, q, alpha, smooth.null, bw, kernel, prior, keep.nulldist, keep.rawdist, keep.falsepos, keep.truepos, keep.errormat, keep.margpar. When evaluate is 'TRUE', a new object of class EBMTP is returned. Else, the updated call is returned. The EBMTP object passed to the update method must have either a non-empty rawdist slot or a non-empty nulldist slot (i.e., must have been called with either 'keep.rawdist=TRUE' or 'keep.nulldist=TRUE').

Additionally, when calling EBupdate for any Type I error rate other than FWER, the typeone argument must be specified (even if the original object did not control FWER). For example, typeone="fdr", would always have to be specified, even if the original object also controlled the FDR. In other words, for all function arguments, it is safest to always assume that you are updating from the EBMTP default function settings, regardless of the original call to the EBMTP function. Currently, the main advantage of the EBupdate method is that it prevents the need for repeated estimation of the test statistics null distribution.

To save on memory, if one knows ahead of time that one will want to compare different choices of bootstrap-based null distribution, then it is both necessary and sufficient to specify 'keep.rawdist=TRUE', as there is no other means of moving between null distributions other than through the non-transformed non-parametric bootstrap distribution. In this case, 'keep.nulldist=FALSE' may be used. Specifically, if an object of class EBMTP contains a non-empty rawdist slot and an empty nulldist slot, then a new null distribution will be generated according to the values of the nulldist= argument in the original call to EBMTP or any additional specifications in the call to update. On the other hand, if one knows that one wishes to only update an EBMTP object in ways which do not involve choice of null distribution, then 'keep.nulldist=TRUE' will suffice and 'keep.rawdist' can be set to FALSE (default settings). The original null distribution object will then be used for all subsequent calls to update.

N.B.: Note that keep.rawdist=TRUE is only available for the bootstrap-based resampling methods. The non-null distribution does not exist for the permutation or influence curve multivariate normal null distributions.

ebmtp2mtp : coersion method for converting objects of class EBMTP to objects of class MTP. Slots common to both objects are taken from the object of class EBMTP and used to create a new object of class MTP. Once an object of class MTP is created, one may use the method update to perform resampling-based multiple testing (as would have been done with calls to MTP) without the need for repeated resampling.

Author(s)

Houston N. Gilbert, based on the original MTP class and method definitions written by Katherine S. Pollard

References

H.N. Gilbert, K.S. Pollard, M.J. van der Laan, and S. Dudoit (2009). Resampling-based multiple hypothesis testing with applications to genomics: New developments in R/Bioconductor package multtest. *Journal of Statistical Software* (submitted). Temporary URL: http://www.stat.berkeley.edu/~houston/JSSNullDistEBMTP.pdf.

Y. Benjamini and Y. Hochberg (2000). On the adaptive control of the false discovery rate in multiple testing with independent statistics. *J. Behav. Educ. Statist*. Vol 25: 60-83.

Y. Benjamini, A. M. Krieger and D. Yekutieli (2006). Adaptive linear step-up procedures that control the false discovery rate. *Biometrika*. Vol. 93: 491-507.

M.J. van der Laan, M.D. Birkner, and A.E. Hubbard (2005). Empirical Bayes and Resampling Based Multiple Testing Procedure Controlling the Tail Probability of the Proportion of False Positives. Statistical Applications in Genetics and Molecular Biology, 4(1). http://www.bepress.com/sagmb/vol4/iss1/art29/

S. Dudoit and M.J. van der Laan. Multiple Testing Procedures and Applications to Genomics. Springer Series in Statistics. Springer, New York, 2008.

S. Dudoit, H. N. Gilbert, and M. J. van der Laan (2008). Resampling-based empirical Bayes multiple testing procedures for controlling generalized tail probability and expected value error rates: Focus on the false discovery rate and simulation study. *Biometrical Journal*, 50(5):716-44. http://www.stat.berkeley.edu/~houston/BJMCPSupp/BJMCPSupp.html.

H.N. Gilbert, M.J. van der Laan, and S. Dudoit. Joint multiple testing procedures for graphical model selection with applications to biological networks. Technical report, U.C. Berkeley Division of Biostatistics Working Paper Series, April 2009. URL http://www.bepress.com/ucbbiostat/paper245.

See Also

EBMTP, EBMTP-methods, MTP, MTP-methods, [-methods, as.list-methods, print-methods, plot-methods, summary-methods, mtp2ebmtp, ebmtp2mtp

Examples

```
## See EBMTP function: ? EBMTP
```

`fwer2gfwer` *Function to compute augmentation MTP adjusted p-values*

Description

Augmentation multiple testing procedures (AMTPs) to control the generalized family-wise error rate (gFWER), the tail probability of the proportion of false positives (TPPFP), and false discovery rate (FDR) based on any initial procudeure controlling the family-wise error rate (FWER). AMTPs are obtained by adding suitably chosen null hypotheses to the set of null hypotheses already rejected by an initial FWER-controlling MTP. A function for control of FDR given any TPPFP controlling procedure is also provided.

Usage

```
fwer2gfwer(adjp, k = 0)

fwer2tppfp(adjp, q = 0.05)

fwer2fdr(adjp, method = "both", alpha = 0.05)
```

Arguments

`adjp`	Numeric vector of adjusted p-values from any FWER-controlling procedure.
`k`	Maximum number of false positives.
`q`	Maximum proportion of false positives.
`method`	Character string indicating which FDR controlling method should be used. The options are "conservative" for a conservative, general method, "restricted" for a less conservative, but restricted method, or "both" (default) for both.
`alpha`	Nominal level for an FDR controlling procedure (can be a vector of levels).

Details

The gFWER and TPPFP functions control Type I error rates defined as tail probabilities for functions g(Vn,Rn) of the numbers of Type I errors (Vn) and rejected hypotheses (Rn). The gFWER and TPPFP correspond to the special cases g(Vn,Rn)=Vn (number of false positives) and g(Vn,Rn)=Vn/Rn (proportion of false positives among the rejected hypotheses), respectively.

Adjusted p-values for an AMTP are simply shifted versions of the adjusted p-values of the original FWER-controlling MTP. For control of gFWER (Pr(Vn>k)), for example, the first `k` adjusted p-values are set to zero and the remaining p-values are the adjusted p-values of the FWER-controlling MTP shifted by k. One can therefore build on the large pool of available FWER-controlling procedures, such as the single-step and step-down maxT and minP procedures.

Given a FWER-controlling MTP, the FDR can be conservatively controlled at level `alpha` by considering the corresponding TPPFP AMTP with `q=alpha/2` at level `alpha/2`, so that Pr(Vn/Rn>alpha/2)<=alpha/2. A less conservative procedure (`general=FALSE`) is obtained by using an AMTP controlling the

TPPFP with q=1-sqrt(1-alpha) at level 1-sqrt(1-alpha), so that Pr(Vn/Rn>1-sqrt(1-alpha))<=1-sqrt(1-alpha). The first, more general method can be used with any procedure that asymptotically controls FWER. The second, less conservative method requires the following additional assumptions: (i) the true alternatives are asymptotically always rejected by the FWER-controlling procedure, (ii) the limit of the FWER exists, and (iii) the FWER-controlling procedure provides exact asymptotic control. See http://www.bepress.com/sagmb/vol3/iss1/art15/ for more details. The method implemented in fwer2fdr for computing rejections simply uses the TPPFP AMTP fwer2tppfp with q=alpha/2 (or 1-sqrt(1-alpha)) and rejects each hypothesis for which the TPPFP adjusted p-value is less than or equal to alpha/2 (or 1-sqrt(1-alpha)). The adjusted p-values are based directly on the FWER adjusted p-values, so that very occasionally a hypothesis will have the indicator that it is rejected in the matrix of rejections, but the adjusted p-value will be slightly greater than the nominal level. The opposite might also occur occasionally.

Value

For fwer2gfwer and fwer2tppfp, a numeric vector of AMTP adjusted p-values. For fwer2fdr, a list with two components: (i) a numeric vector (or a length(adjp) by 2 matrix if method="both") of adjusted p-values for each hypothesis, (ii) a length(adjp) by length(alpha) matrix (or length(adjp) by length(alpha) by 2 array if method="both") of indicators of whether each hypothesis is rejected at each value of the argument alpha.

Author(s)

Katherine S. Pollard with design contributions from Sandrine Dudoit and Mark J. van der Laan.

References

M.J. van der Laan, S. Dudoit, K.S. Pollard (2004), Augmentation Procedures for Control of the Generalized Family-Wise Error Rate and Tail Probabilities for the Proportion of False Positives, Statistical Applications in Genetics and Molecular Biology, 3(1). http://www.bepress.com/sagmb/vol3/iss1/art15/

M.J. van der Laan, S. Dudoit, K.S. Pollard (2004), Multiple Testing. Part II. Step-Down Procedures for Control of the Family-Wise Error Rate, Statistical Applications in Genetics and Molecular Biology, 3(1). http://www.bepress.com/sagmb/vol3/iss1/art14/

S. Dudoit, M.J. van der Laan, K.S. Pollard (2004), Multiple Testing. Part I. Single-Step Procedures for Control of General Type I Error Rates, Statistical Applications in Genetics and Molecular Biology, 3(1). http://www.bepress.com/sagmb/vol3/iss1/art13/

Katherine S. Pollard and Mark J. van der Laan, "Resampling-based Multiple Testing: Asymptotic Control of Type I Error and Applications to Gene Expression Data" (June 24, 2003). U.C. Berkeley Division of Biostatistics Working Paper Series. Working Paper 121. http://www.bepress.com/ucbbiostat/paper121

See Also

MTP, MTP-class, MTP-methods, mt.minP, mt.maxT

Examples

```
data<-matrix(rnorm(200),nr=20)
group<-c(rep(0,5),rep(1,5))
fwer.mtp<-MTP(X=data,Y=group)
fwer.adjp<-fwer.mtp@adjp
gfwer.adjp<-fwer2gfwer(adjp=fwer.adjp,k=c(1,5,10))
compare.gfwer<-cbind(fwer.adjp,gfwer.adjp)
mt.plot(adjp=compare.gfwer,teststat=fwer.mtp@statistic,proc=c("gFWER(0)","gFWER(1)","gFWER(5)","gFWER(10)"),co
title("Comparison of Single-step MaxT gFWER Controlling Methods")
```

`get.index` *Function to compute indices for ordering hypotheses in Package 'multtest'*

Description

The hypotheses tested in a multiple testing procedure (MTP), can be ordered based on the output of that procedure. This function orders hypotheses based on adjusted p-values, then unadjusted p-values (to break ties in adjusted p-values), and finally test statistics (to break remaining ties).

Usage

```
get.index(adjp, rawp, stat)
```

Arguments

`adjp`	Numeric vector of adjusted p-values.
`rawp`	Numeric vector of unadjusted ("raw") marginal p-values.
`stat`	Numeric vector of test statistics.

Value

Numeric vector of indices so that the hypotheses can be ordered accroding to significance (smallest p-values and largest test statistics first). This function is used in the plot method for objects of class `MTP` to order adjusted p-values for graphical summaries. The summary method for objects of class `MTP` will return these indices as its second component.

Author(s)

Katherine S. Pollard

See Also

`MTP`, `plot,MTP,ANY-method`, `summary,MTP-method`

Examples

```
data<-matrix(rnorm(200),nr=20)
mtp<-MTP(X=data,test="t.onesamp")
index<-get.index(adjp=mtp@adjp,rawp=mtp@rawp,stat=mtp@statistic)
mtp@statistic[index]
mtp@estimate[index]
apply(data[index,],1,mean)
```

---

golub *Gene expression dataset from Golub et al. (1999)*

Description

Gene expression data (3051 genes and 38 tumor mRNA samples) from the leukemia microarray study of Golub et al. (1999). Pre-processing was done as described in Dudoit et al. (2002). The R code for pre-processing is available in the file ../doc/golub.R.

Usage

```
data(golub)
```

Value

golub	matrix of gene expression levels for the 38 tumor mRNA samples, rows correspond to genes (3051 genes) and columns to mRNA samples.
golub.cl	numeric vector indicating the tumor class, 27 acute lymphoblastic leukemia (ALL) cases (code 0) and 11 acute myeloid leukemia (AML) cases (code 1).
golub.gnames	a matrix containing the names of the 3051 genes for the expression matrix golub. The three columns correspond to the gene index, ID, and Name, respectively.

Source

Golub et al. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring, *Science*, Vol. 286:531-537.
http://www-genome.wi.mit.edu/MPR/ .

References

S. Dudoit, J. Fridlyand, and T. P. Speed (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, Vol. 97, No. 457, p. 77–87.

Hsets — *Functions for generating guessed sets of true null hypotheses in empirical Bayes resampling-based multiple hypothesis testing*

Description

These functions are called internally by the main user-level function EBMTP. They are used for estimating local q-values, generating guessed sets of true null hypotheses, and applying these results to function closures defining the choice of type I error rate (FWER, gFWER, TPPFP, and FDR).

Usage

```
Hsets(Tn, nullmat, bw, kernel, prior, B, rawp)

ABH.h0(rawp)

G.VS(V, S = NULL, tp = TRUE, bound)
```

Arguments

Tn	The vector of observed test statistics.
nullmat	The matrix of null test statistics obtained either through null transformation of the bootstrap distribution or by sampling from an appropriate multivariate normal distribution (when `nulldist='ic'`.)
bw	A character string argument to `density` indicating the smoothing bandwidth to be used during kernel density estimation. Default is 'nrd'.
kernel	A character string argument to `density` specifying the smoothing kernel to be used. Default is 'gaussian'.
prior	Character string indicating which choice of prior probability to use for estimating local q-values (i.e., the posterior probabilities of a null hypothesis being true given the value of its corresponding test statistic). Default is 'conservative', in which case the prior is set to its most conservative value of 1, meaning that all hypotheses are assumed to belong to the set of true null hypotheses. Other options include 'ABH' for the adaptive Benjamini-Hochberg estimator of the number/proportion of true null hypotheses, and 'EBLQV' for the empirical Bayes local q-value value estimator of the number/proportion of true null hypotheses. If 'EBLQV', the estimator of the prior probability is taken to be the sum of the estimated local q-values divided by the number of tests. Relaxing the prior may result in more rejections, albeit at a cost of type I error control under certain conditions. See references.
B	The number of bootstrap iterations (i.e. how many resampled data sets) or the number of samples from the multivariate normal distribution (if `nulldist='ic'`). Can be reduced to increase the speed of computation, at a cost to precision. Default is 1000.
rawp	A vector of raw (unadjusted) p-values obtained bootstrap-based or influence curve null distribution.

V	A matrix of the numbers of guessed false positives for each cut-off, i.e., observed value of a test statistic, within each sample in B.
S	A matrix of the numbers of guessed true positives for each cut-off, i.e., observed value of a test statistic, within each sample in B.
tp	Logical indicator which is TRUE if type I error rate is a tail probability error rate and FALSE is if it is an expected value error rate.
bound	If a tail probability error rate, the bound to be placed on function of guessed false positives and guessed true positives. For, 'fwer', equal to 0; 'gfwer', equal to 'k'; and tppfp, equal to 'q'.

Details

The most important object to be returned from the function Hsets is a matrix of indicators, i.e., Bernoulli realizations of the estimated local q-values, taking the value of 1 if the hypothesis is guessed as belonging to the set of true null hypotheses and 0 otherwise (guessed true alternative). Realizations of these probabilities are generated with a call to rbinom, meaning that this function will set the RNG seed forward another B*(the number of hypotheses) places. This matrix, with rows equal to the number of hypotheses and columns the number of (bootstrap or multivariate normal) samples is used to subset the matrix of null test statistics and the vector of observed test statistics at each round of (re)sampling into samples of statistics guessed as belonging to the sets of true null and true alternative hypotheses, respectively. Using the values of the observed test statistics themselves as cut-offs, the numbers of guessed false positives and (if applicable) guessed true positives can be counted and eventually used to estimate the distribution of a type I error rate characterized by the closure returned from G.VS. Counting of guessed false positives and guessed true positives is performed in C through the function VScount.

Value

For the function Hsets, a list with the following elements:

Hsets.mat	A matrix of numeric indicators with rows equal to the number of test (hypotheses, typically nrow(X)) and columns the number of samples of null test statistics, B. Values of one indicate hypotheses guessed as belonging to the set of true null hypotheses based on the value of their corresponding test statistic. Values of zero correspond to hypotheses guesses as belonging to the set of true alternative hypotheses.
EB.h0M	The estimated proportion of true null hypotheses as determined by nonparametric density estimation. This value is the sum of the estimated local q-values divided by the total number of tests (hypotheses).
prior	The value of the prior applied to the local q-value function. If 'conservative', the prior is set to one. Otherwise, the prior is the value obtained from the estimator of the adaptive Benjamini-Hochberg procedure (if prior is 'ABH') or from density estimation (if prior is 'EBLQV').
pn.out	The vector of estimated local q-values. This vector is returned in the lqv slot of objects of class EBMTP.

For the function ABH.h0, the estimated number of true null hypotheses using the estimator from the linear step-up adaptive Benjamini-Hochberg procedure.

For the function `G.VS`, a closure which accepts as arguments the matrices of guessed false positive and true positives (if applicable) and applies the appropriate function defining the desired type I error rate.

Author(s)

Houston N. Gilbert

References

H.N. Gilbert, K.S. Pollard, M.J. van der Laan, and S. Dudoit (2009). Resampling-based multiple hypothesis testing with applications to genomics: New developments in R/Bioconductor package multtest. *Journal of Statistical Software* (submitted). Temporary URL: http://www.stat.berkeley.edu/~houston/JSSNullDistEBMTP.pdf.

Y. Benjamini and Y. Hochberg (2000). On the adaptive control of the false discovery rate in multiple testing with independent statistics. *J. Behav. Educ. Statist*. Vol 25: 60-83.

Y. Benjamini, A.M. Krieger and D. Yekutieli (2006). Adaptive linear step-up procedures that control the false discovery rate. *Biometrika*. Vol. 93: 491-507.

M.J. van der Laan, M.D. Birkner, and A.E. Hubbard (2005). Empirical Bayes and Resampling Based Multiple Testing Procedure Controlling the Tail Probability of the Proportion of False Positives. Statistical Applications in Genetics and Molecular Biology, 4(1). http://www.bepress.com/sagmb/vol4/iss1/art29/

S. Dudoit and M.J. van der Laan. Multiple Testing Procedures and Applications to Genomics. Springer Series in Statistics. Springer, New York, 2008.

S. Dudoit, H.N. Gilbert, and M.J. van der Laan (2008). Resampling-based empirical Bayes multiple testing procedures for controlling generalized tail probability and expected value error rates: Focus on the false discovery rate and simulation study. *Biometrical Journal*, 50(5):716-44. http://www.stat.berkeley.edu/~houston/BJMCPSupp/BJMCPSupp.html.

H.N. Gilbert, M.J. van der Laan, and S. Dudoit. Joint multiple testing procedures for graphical model selection with applications to biological networks. Technical report, U.C. Berkeley Division of Biostatistics Working Paper Series, April 2009. URL http://www.bepress.com/ucbbiostat/paper245.

See Also

EBMTP, EBMTP-class, EBMTP-methods

Examples

```
set.seed(99)
data<-matrix(rnorm(90),nr=9)
```

```
group<-c(rep(1,5),rep(0,5))

#EB fwer control with centered and scaled bootstrap null distribution
#(B=100 for speed)
eb.m1<-EBMTP(X=data,Y=group,alternative="less",B=100,method="common.cutoff")
print(eb.m1)
summary(eb.m1)
par(mfrow=c(2,2))
plot(eb.m1,top=9)

abh <- ABH.h0(eb.m1@rawp)
abh

eb.m2 <- EBupdate(eb.m1,prior="ABH")
eb.m2@prior
```

---

meanX *Functions to create test statistic closures and apply them to data*

Description

The package multtest uses closures in the function MTP to compute test statistics. The closure used depends on the value of the argument test. These functions create the closures for different tests, given any additional variables, such as outcomes or covariates. The function get.Tn calls wapply to apply one of these closures to observed data (and possibly weights).

One exception for how test statistics are calculated in multtest involve tests of correlation parameters, where the change of dimensionality between the p variables in X and the p-choose-2 hypotheses corresponding to the number of pairwise correlations presents a challenge. In this case, the test statistics are calculated directly in corr.Tn and returned in a manner similar to the test statistic function closures. No resampling is done either, since the null distribution for tests of correlation parameters are only implemented when nulldist='ic'. Details are given below.

Usage

```
meanX(psi0 = 0, na.rm = TRUE, standardize = TRUE,
alternative = "two.sided", robust = FALSE)

diffmeanX(label, psi0 = 0, var.equal = FALSE, na.rm = TRUE,
standardize = TRUE, alternative = "two.sided", robust = FALSE)

FX(label, na.rm = TRUE, robust = FALSE)

blockFX(label, na.rm = TRUE, robust = FALSE)

twowayFX(label, na.rm = TRUE, robust = FALSE)
```

```
lmX(Z = NULL, n, psi0 = 0, na.rm = TRUE, standardize = TRUE,
alternative = "two.sided", robust = FALSE)

lmY(Y, Z = NULL, n, psi0 = 0, na.rm = TRUE, standardize = TRUE,
alternative = "two.sided", robust = FALSE)

coxY(surv.obj, strata = NULL, psi0 = 0, na.rm = TRUE, standardize = TRUE,
alternative = "two.sided", init = NULL, method = "efron")

get.Tn(X, stat.closure, W = NULL)

corr.Tn(X, test, alternative, use = "pairwise")
```

Arguments

X	A matrix, data.frame or ExpressionSet containing the raw data. In the case of an ExpressionSet, `exprs(X)` is the data of interest and `pData(X)` may contain outcomes and covariates of interest. For currently implemented tests, one hypothesis is tested for each row of the data.
W	A vector or matrix containing non-negative weights to be used in computing the test statistics. If a matrix, `W` must be the same dimension as `X` with one weight for each value in `X`. If a vector, `W` may contain one weight for each observation (i.e. column) of `X` or one weight for each variable (i.e. row) of `X`. In either case, the weights are duplicated apporpraiately. Weighted f-tests are not available. Default is 'NULL'.
label	A vector containing the class labels for t- and f-tests. For the `blockFX` function, observations are divided into `l` blocks of `n/l` observations. Within each block there may be `k` groups with `k>2`. For this test, there is only one observation per block*group combination. The labels (and corresponding rows of `Z` and columns of `X` and `W`) must be ordered by block and within each block ordered by group. Groups must be labeled with integers `1,...,k`. For the `twowayFX` function, observations are divided into `l` blocks. Within each block there may be `k` groups with `k>2`. There must be more than one observation per group*block combination for this test. The labels (and corresponding rows of `Z` and columns of `X` and `W`) must be ordered by block and within each block ordered by group. Groups must be labeled with integers `1,...,k`.
Y	A vector or factor containing the outcome of interest for linear models. This may be a continuous or polycotomous dependent variable.
surv.object	A survival object as returned by the `Surv` function, to be used as response in `coxY`.
Z	A vector, factor, or matrix containing covariate data to be used in the linear regression models. Each variable should be in one column.
strata	A vector, factor, or matrix containing covariate data to be used in the Cox regression models. Covariate data will be converted to a factor variable (via the `strata` function) for use in the `coxph` function. Each variable should be in one column.
n	The sample size, e.g. `length(Y)` or `nrow(Z)`.

psi0
: Hypothesized null value for the parameter of interest (e.g. mean or difference in means), typically zero (default).

var.equal
: Indicator of whether to use t-statistics that assume equal variance in the two groups when computing the denominator of the test statistics.

na.rm
: Logical indicating whether to remove observations with an NA. Default is 'TRUE'.

standardize
: Logical indicating whether to use the standardized version of the test statistics (usual t-statistics are standardized). Default is 'TRUE'.

alternative
: Character string indicating the alternative hypotheses, by default 'two.sided'. For one-sided tests, use 'less' or 'greater' for null hypotheses of 'greater than or equal' (i.e. alternative is 'less') and 'less than or equal', respectively.

robust
: Logical indicating whether to use robust versions of the test statistics.

init
: Vector of initial values of the iteration in coxY function, as used in coxph in the survival package. Default initial value is zero for all variables (init=NULL).

method
: A character string specifying the method for tie handling in coxY function, as used in coxph in the survival package. Default is "efron".

test
: For corr.Tn, a character string of either 't.cor' or 'z.cor' indicating whether t-statistics or Fisher's z-statistics are to be calculated when probing hypotheses involving correlation parameters.

use
: Similar to the options in cor, a character string giving a method for computing covariances in the presence of missing values. Default is 'pairwise', which allows for the covariance/correlation matrix to be calculated using the most information possible when NAs are present.

Details

The use of closures, in the style of the genefilter package, allows uniform data input for all MTPs and facilitates the extension of the package's functionality by adding, for example, new types of test statistics. Specifically, for each value of the MTP argument test, a closure is defined which consists of a function for computing the test statistic (with only two arguments, a data vector x and a corresponding weight vector w, with default value of NULL) and its enclosing environment, with bindings for relevant additional arguments. These arguments may include null values psi0, outcomes (Y, label, surv.object), and covariates Z. The vectors x and w are rows of the matrices X and W.

In the MTP function, the closure is first used to compute the vector of observed test statistics, and then, in each bootstrap iteration, to produce the estimated joint null distribution of the test statistics. In both cases, the function get.Tn is used to apply the closure to rows of the matrices of data (X) and weights (W). Thus, new test statistics can be added to multtest package by simply defining a new closure and adding a corresponding value for the test argument to the MTP function.

As mentioned above, one exception made to the closure rule in multtest was done for the case of tests involving correlation parameters (i.e., when test='t.cor' or test='z.cor'). In particular, the change of dimension between the number of variables in X and the number of hypotheses corresponding to all pairwise correlation parameters presented a challenge. In this setting, a 'closure-like' function was written which returns choose(dim(X)[2],2) test statistics stored in a matrix obs described below. No resampling methods are available for 't.cor' and 'z.cor', as their only current available null distribution is based on influence curves (nulldist='ic'), meaning that the test statistics null distribution is sampled directly from an appropriate multivariate normal distribution. In this manner, the data are used to calculate test statistics and null distribution estimates of the appropriate length and dimension, with sidedness correctly accounted for. With care, these objects for tests of correlation can then be integrated into the rest of the (modular) `multtest` functionality to perform multiple testing using other available argument options in the functions `MTP` or `EBMTP`.

Value

For `meanX`, `diffmeanX`, `FX`, `blockFX`, `twowayFX`, `lmX`, `lmY`, and `coxY`, a closure consisting of a function for computing test statistics and its enclosing environment. For `get.Tn` and `corr.Tn`, the observed test statistics stored in a matrix `obs` with numerator (possibly absolute value or negative, depending on the value of `alternative`) in the first row, denominator in the second row, and a 1 or -1 in the third row (depending on the value of alternative). The vector of observed test statistics is obs[1,]*obs[3,]/obs[2,].

Author(s)

Katherine S. Pollard, Houston N. Gilbert, and Sandra Taylor, with design contributions from Duncan Temple Lang, Sandrine Dudoit and Mark J. van der Laan

See Also

`MTP`, `get.Tn`, `wapply`, `boot.resample`

Examples

```
data<-matrix(rnorm(200),nr=20)
#one-sample t-statistics
ttest<-meanX(psi0=0,na.rm=TRUE,standardize=TRUE,alternative="two.sided",robust=FALSE)
obs<-wapply(data,1,ttest,W=NULL)
statistics<-obs[1,]*obs[3,]/obs[2,]
statistics

#for tests of correlation parameters,
#note change of dimension compared to dim(data),
#function calculate statistics directly in same form as above
obs <- corr.Tn(data,test="t.cor",alternative="greater")
dim(obs)
statistics<-obs[1,]*obs[3,]/obs[2,]
length(statistics)

#two-way F-statistics
FData <- matrix(rnorm(5*60),nr=5)
label<-rep(c(rep(1,10), rep(2,10), rep(3,10)),2)
twowayf<-twowayFX(label)
obs<-wapply(FData,1,twowayf,W=NULL)
statistics<-obs[1,]*obs[3,]/obs[2,]
statistics
```

mt.maxT *Step-down maxT and minP multiple testing procedures*

Description

These functions compute permutation adjusted p-values for step-down multiple testing procedures described in Westfall & Young (1993).

Usage

```
mt.maxT(X,classlabel,test="t",side="abs",fixed.seed.sampling="y",B=10000,na=.mt.naNUM,nonpara="n")
mt.minP(X,classlabel,test="t",side="abs",fixed.seed.sampling="y",B=10000,na=.mt.naNUM,nonpara="n")
```

Arguments

`X` A data frame or matrix, with m rows corresponding to variables (hypotheses) and n columns to observations. In the case of gene expression data, rows correspond to genes and columns to mRNA samples. The data can be read using `read.table`.

`classlabel` A vector of integers corresponding to observation (column) class labels. For k classes, the labels must be integers between 0 and $k-1$. For the `blockf` test option, observations may be divided into n/k blocks of k observations each. The observations are ordered by block, and within each block, they are labeled using the integers 0 to $k-1$.

`test` A character string specifying the statistic to be used to test the null hypothesis of no association between the variables and the class labels.
If `test="t"`, the tests are based on two-sample Welch t-statistics (unequal variances).
If `test="t.equalvar"`, the tests are based on two-sample t-statistics with equal variance for the two samples. The square of the t-statistic is equal to an F-statistic for $k=2$.
If `test="wilcoxon"`, the tests are based on standardized rank sum Wilcoxon statistics.
If `test="f"`, the tests are based on F-statistics.
If `test="pairt"`, the tests are based on paired t-statistics. The square of the paired t-statistic is equal to a block F-statistic for $k=2$.
If `test="blockf"`, the tests are based on F-statistics which adjust for block differences (cf. two-way analysis of variance).

`side` A character string specifying the type of rejection region.
If `side="abs"`, two-tailed tests, the null hypothesis is rejected for large absolute values of the test statistic.
If `side="upper"`, one-tailed tests, the null hypothesis is rejected for large values of the test statistic.
If `side="lower"`, one-tailed tests, the null hypothesis is rejected for small values of the test statistic.

fixed.seed.sampling
: If fixed.seed.sampling="y", a fixed seed sampling procedure is used, which may double the computing time, but will not use extra memory to store the permutations. If fixed.seed.sampling="n", permutations will be stored in memory. For the blockf test, the option n was not implemented as it requires too much memory.

B
: The number of permutations. For a complete enumeration, B should be 0 (zero) or any number not less than the total number of permutations.

na
: Code for missing values (the default is .mt.naNUM=--93074815.62). Entries with missing values will be ignored in the computation, i.e., test statistics will be based on a smaller sample size. This feature has not yet fully implemented.

nonpara
: If nonpara="y", nonparametric test statistics are computed based on ranked data.
If nonpara="n", the original data are used.

Details

These functions compute permutation adjusted p-values for the step-down maxT and minP multiple testing procedures, which provide strong control of the family-wise Type I error rate (FWER). The adjusted p-values for the minP procedure are defined in equation (2.10) p. 66 of Westfall & Young (1993), and the maxT procedure is discussed p. 50 and 114. The permutation algorithms for estimating the adjusted p-values are given in Ge et al. (In preparation). The procedures are for the simultaneous test of m null hypotheses, namely, the null hypotheses of no association between the m variables corresponding to the rows of the data frame X and the class labels classlabel. For gene expression data, the null hypotheses correspond to no differential gene expression across mRNA samples.

Value

A data frame with components

index
: Vector of row indices, between 1 and nrow(X), where rows are sorted first according to their adjusted p-values, next their unadjusted p-values, and finally their test statistics.

teststat
: Vector of test statistics, ordered according to index. To get the test statistics in the original data order, use teststat[order(index)].

rawp
: Vector of raw (unadjusted) p-values, ordered according to index.

adjp
: Vector of adjusted p-values, ordered according to index.

plower
: For mt.minP function only, vector of "adjusted p-values", where ties in the permutation distribution of the successive minima of raw p-values with the observed p-values are counted only once. Note that procedures based on plower do not control the FWER. Comparison of plower and adjp gives an idea of the discreteness of the permutation distribution. Values in plower are ordered according to index.

Author(s)

Yongchao Ge, <yongchao.ge@mssm.edu>,
Sandrine Dudoit, http://www.stat.berkeley.edu/~sandrine.

References

S. Dudoit, J. P. Shaffer, and J. C. Boldrick (Submitted). Multiple hypothesis testing in microarray experiments.

Y. Ge, S. Dudoit, and T. P. Speed. Resampling-based multiple testing for microarray data hypothesis, Technical Report \#633 of UCB Stat. http://www.stat.berkeley.edu/~gyc

P. H. Westfall and S. S. Young (1993). *Resampling-based multiple testing: Examples and methods for p-value adjustment*. John Wiley \& Sons.

See Also

mt.plot, mt.rawp2adjp, mt.reject, mt.sample.teststat, mt.teststat, golub.

Examples

```
# Gene expression data from Golub et al. (1999)
# To reduce computation time and for illustrative purposes, we condider only
# the first 100 genes and use the default of B=10,000 permutations.
# In general, one would need a much larger number of permutations
# for microarray data.

data(golub)
smallgd<-golub[1:100,]
classlabel<-golub.cl

# Permutation unadjusted p-values and adjusted p-values
# for maxT and minP procedures with Welch t-statistics
resT<-mt.maxT(smallgd,classlabel)
resP<-mt.minP(smallgd,classlabel)
rawp<-resT$rawp[order(resT$index)]
teststat<-resT$teststat[order(resT$index)]

# Plot results and compare to Bonferroni procedure
bonf<-mt.rawp2adjp(rawp, proc=c("Bonferroni"))
allp<-cbind(rawp, bonf$adjp[order(bonf$index),2], resT$adjp[order(resT$index)],resP$adjp[order(resP$index)])

mt.plot(allp, teststat, plottype="rvsa", proc=c("rawp","Bonferroni","maxT","minP"),leg=c(0.7,50),lty=1,col=1:4,
mt.plot(allp, teststat, plottype="pvsr", proc=c("rawp","Bonferroni","maxT","minP"),leg=c(60,0.2),lty=1,col=1:4,
mt.plot(allp, teststat, plottype="pvst", proc=c("rawp","Bonferroni","maxT","minP"),leg=c(-6,0.6),pch=16,col=1:4

# Permutation adjusted p-values for minP procedure with F-statistics (like equal variance t-statistics)
mt.minP(smallgd,classlabel,test="f",fixed.seed.sampling="n")

# Note that the test statistics used in the examples below are not appropriate
# for the Golub et al. data. The sole purpose of these examples is to
# demonstrate the use of the mt.maxT and mt.minP functions.

# Permutation adjusted p-values for maxT procedure with paired t-statistics
classlabel<-rep(c(0,1),19)
mt.maxT(smallgd,classlabel,test="pair")
```

```
# Permutation adjusted p-values for maxT procedure with block F-statistics
classlabel<-rep(0:18,2)
mt.maxT(smallgd,classlabel,test="blockf",side="upper")
```

---

mt.plot *Plotting results from multiple testing procedures*

Description

This function produces a number of graphical summaries for the results of multiple testing procedures and their corresponding adjusted p-values.

Usage

```
mt.plot(adjp, teststat, plottype="rvsa", logscale=FALSE, alpha=seq(0, 1, length = 100), proc, leg=c(0,
```

Arguments

`adjp`	A matrix of adjusted p-values, with rows corresponding to hypotheses (genes) and columns to multiple testing procedures. This matrix could be obtained from the functions `mt.maxT`, `mt.minP`, or `mt.rawp2adjp`.
`teststat`	A vector of test statistics for each of the hypotheses. This vector could be obtained from the functions `mt.teststat`, `mt.maxT`, or `mt.minP`.
`plottype`	A character string specifying the type of graphical summary for the results of the multiple testing procedures. If `plottype="rvsa"`, the number of rejected hypotheses is plotted against the nominal Type I error rate for each of the procedures given in `proc`. If `plottype="pvsr"`, the ordered adjusted p-values are plotted for each of the procedures given in `proc`. This can be viewed as a plot of the Type I error rate against the number of rejected hypotheses. If `plottype="pvst"`, the adjusted p-values are plotted against the test statistics for each of the procedures given in `proc`. If `plottype="pvsi"`, the adjusted p-values are plotted for each of the procedures given in `proc` using the original data order.
`logscale`	A logical variable for the `pvst` and `pvsi` plots. If `logscale` is `TRUE`, the negative decimal logarithms of the adjusted p-values are plotted against the test statistics or gene indices. If `logscale` is `FALSE`, the adjusted p-values are plotted against the test statistics or gene indices.
`alpha`	A vector of nominal Type I error rates for the `rvsa` plot.
`proc`	A vector of character strings containing the names of the multiple testing procedures, to be used in the legend.
`...`	Graphical parameters such as `col`, `lty`, `pch`, and `lwd` may also be supplied as arguments to the function (see `par`).
`leg`	A vector of coordinates for the legend.

Author(s)

Sandrine Dudoit, http://www.stat.berkeley.edu/~sandrine,
Yongchao Ge, <yongchao.ge@mssm.edu>.

References

S. Dudoit, J. P. Shaffer, and J. C. Boldrick (Submitted). Multiple hypothesis testing in microarray experiments.

Y. Ge, S. Dudoit, and T. P. Speed. Resampling-based multiple testing for microarray data hypothesis, Technical Report \#633 of UCB Stat. http://www.stat.berkeley.edu/~gyc

See Also

mt.maxT, mt.minP, mt.rawp2adjp, mt.reject, mt.teststat, golub.

Examples

```
# Gene expression data from Golub et al. (1999)
# To reduce computation time and for illustrative purposes, we condider only
# the first 100 genes and use the default of B=10,000 permutations.
# In general, one would need a much larger number of permutations
# for microarray data.

data(golub)
smallgd<-golub[1:100,]
classlabel<-golub.cl

# Permutation unadjusted p-values and adjusted p-values for maxT procedure
res1<-mt.maxT(smallgd,classlabel)
rawp<-res1$rawp[order(res1$index)]
teststat<-res1$teststat[order(res1$index)]

# Permutation adjusted p-values for simple multiple testing procedures
procs<-c("Bonferroni","Holm","Hochberg","SidakSS","SidakSD","BH","BY")
res2<-mt.rawp2adjp(rawp,procs)

# Plot results from all multiple testing procedures
allp<-cbind(res2$adjp[order(res2$index),],res1$adjp[order(res1$index)])
dimnames(allp)[[2]][9]<-"maxT"
procs<-dimnames(allp)[[2]]
procs[7:9]<-c("maxT","BH","BY")
allp<-allp[,procs]

cols<-c(1:4,"orange","brown","purple",5:6)
ltypes<-c(3,rep(1,6),rep(2,2))

# Ordered adjusted p-values
mt.plot(allp,teststat,plottype="pvsr",proc=procs,leg=c(80,0.4),lty=ltypes,col=cols,lwd=2)
```

```
# Adjusted p-values in original data order
mt.plot(allp,teststat,plottype="pvsi",proc=procs,leg=c(80,0.4),lty=ltypes,col=cols,lwd=2)

# Number of rejected hypotheses vs. level of the test
mt.plot(allp,teststat,plottype="rvsa",proc=procs,leg=c(0.05,100),lty=ltypes,col=cols,lwd=2)

# Adjusted p-values vs. test statistics
mt.plot(allp,teststat,plottype="pvst",logscale=TRUE,proc=procs,leg=c(0,4),pch=ltypes,col=cols)
```

---

mt.rawp2adjp *Adjusted p-values for simple multiple testing procedures*

Description

This function computes adjusted p-values for simple multiple testing procedures from a vector of raw (unadjusted) p-values. The procedures include the Bonferroni, Holm (1979), Hochberg (1988), and Sidak procedures for strong control of the family-wise Type I error rate (FWER), and the Benjamini & Hochberg (1995) and Benjamini & Yekutieli (2001) procedures for (strong) control of the false discovery rate (FDR). The less conservative adaptive Benjamini & Hochberg (2000) and two-stage Benjamini & Hochberg (2006) FDR-controlling procedures are also included.

Usage

```
mt.rawp2adjp(rawp, proc=c("Bonferroni", "Holm", "Hochberg", "SidakSS", "SidakSD",
"BH", "BY","ABH","TSBH"), alpha = 0.05, na.rm = FALSE)
```

Arguments

`rawp` A vector of raw (unadjusted) p-values for each hypothesis under consideration. These could be nominal p-values, for example, from t-tables, or permutation p-values as given in `mt.maxT` and `mt.minP`. If the `mt.maxT` or `mt.minP` functions are used, raw p-values should be given in the original data order, `rawp[order(index)]`.

`proc` A vector of character strings containing the names of the multiple testing procedures for which adjusted p-values are to be computed. This vector should include any of the following: `"Bonferroni"`, `"Holm"`, `"Hochberg"`, `"SidakSS"`, `"SidakSD"`, `"BH"`, `"BY"`, `"ABH"`, `"TSBH"`.

Adjusted p-values are computed for simple FWER- and FDR- controlling procedures based on a vector of raw (unadjusted) p-values by one or more of the following methods:

Bonferroni Bonferroni single-step adjusted p-values for strong control of the FWER.

Holm Holm (1979) step-down adjusted p-values for strong control of the FWER.

Hochberg Hochberg (1988) step-up adjusted p-values for strong control of the FWER (for raw (unadjusted) p-values satisfying the Simes inequality).

SidakSS Sidak single-step adjusted p-values for strong control of the FWER (for positive orthant dependent test statistics).

SidakSD Sidak step-down adjusted p-values for strong control of the FWER (for positive orthant dependent test statistics).

BH Adjusted p-values for the Benjamini & Hochberg (1995) step-up FDR-controlling procedure (independent and positive regression dependent test statistics).

BY Adjusted p-values for the Benjamini & Yekutieli (2001) step-up FDR-controlling procedure (general dependency structures).

ABH Adjusted p-values for the adaptive Benjamini & Hochberg (2000) step-up FDR-controlling procedure. This method ammends the original step-up procedure using an estimate of the number of true null hypotheses obtained from p-values.

TSBH Adjusted p-values for the two-stage Benjamini & Hochberg (2006) step-up FDR-controlling procedure. This method ammends the original step-up procedure using an estimate of the number of true null hypotheses obtained from a first-pass application of `"BH"`. The adjusted p-values are a-dependent, therefore `alpha` must be set in the function arguments when using this procedure.

`alpha` A nominal type I error rate, or a vector of error rates, used for estimating the number of true null hypotheses in the two-stage Benjamini & Hochberg procedure (`"TSBH"`). Default is 0.05.

`na.rm` An option for handling `NA` values in a list of raw p-values. If `FALSE`, the number of hypotheses considered is the length of the vector of raw p-values. Otherwise, if `TRUE`, the number of hypotheses is the number of raw p-values which were not `NA`s.

Value

A list with components:

`adjp` A matrix of adjusted p-values, with rows corresponding to hypotheses and columns to multiple testing procedures. Hypotheses are sorted in increasing order of their raw (unadjusted) p-values.

`index` A vector of row indices, between 1 and `length(rawp)`, where rows are sorted according to their raw (unadjusted) p-values. To obtain the adjusted p-values in the original data order, use `adjp[order(index),]`.

`h0.ABH` The estimate of the number of true null hypotheses as proposed by Benjamini & Hochberg (2000) used when computing adjusted p-values for the `"ABH"` procedure (see Dudoit et al., 2007).

`h0.TSBH` The estimate (or vector of estimates) of the number of true null hypotheses as proposed by Benjamini et al. (2006) when computing adjusted p-values for the `"TSBH"` procedure. (see Dudoit et al., 2007).

Author(s)

Sandrine Dudoit, http://www.stat.berkeley.edu/~sandrine,
Yongchao Ge, <yongchao.ge@mssm.edu>,

Houston Gilbert, http://www.stat.berkeley.edu/~houston.

References

Y. Benjamini and Y. Hochberg (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J. R. Statist. Soc. B*. Vol. 57: 289-300.

Y. Benjamini and Y. Hochberg (2000). On the adaptive control of the false discovery rate in multiple testing with independent statistics. *J. Behav. Educ. Statist*. Vol 25: 60-83.

Y. Benjamini and D. Yekutieli (2001). The control of the false discovery rate in multiple hypothesis testing under dependency. *Annals of Statistics*. Vol. 29: 1165-88.

Y. Benjamini, A. M. Krieger and D. Yekutieli (2006). Adaptive linear step-up procedures that control the false discovery rate. *Biometrika*. Vol. 93: 491-507.

S. Dudoit, J. P. Shaffer, and J. C. Boldrick (2003). Multiple hypothesis testing in microarray experiments. *Statistical Science*. Vol. 18: 71-103.

S. Dudoit, H. N. Gilbert, and M. J. van der Laan (2008). Resampling-based empirical Bayes multiple testing procedures for controlling generalized tail probability and expected value error rates: Focus on the false discovery rate and simulation study. *Biometrical Journal*, 50(5):716-44. http://www.stat.berkeley.edu/~houston/BJMCPSupp/BJMCPSupp.html.

Y. Ge, S. Dudoit, and T. P. Speed (2003). Resampling-based multiple testing for microarray data analysis. *TEST*. Vol. 12: 1-44 (plus discussion p. 44-77).

Y. Hochberg (1988). A sharper Bonferroni procedure for multiple tests of significance, *Biometrika*. Vol. 75: 800-802.

S. Holm (1979). A simple sequentially rejective multiple test procedure. *Scand. J. Statist.*. Vol. 6: 65-70.

See Also

mt.maxT, mt.minP, mt.plot, mt.reject, golub.

Examples

```
# Gene expression data from Golub et al. (1999)
# To reduce computation time and for illustrative purposes, we condider only
# the first 100 genes and use the default of B=10,000 permutations.
# In general, one would need a much larger number of permutations
# for microarray data.

data(golub)
smallgd<-golub[1:100,]
```

```
classlabel<-golub.cl

# Permutation unadjusted p-values and adjusted p-values for maxT procedure
res1<-mt.maxT(smallgd,classlabel)
rawp<-res1$rawp[order(res1$index)]

# Permutation adjusted p-values for simple multiple testing procedures
procs<-c("Bonferroni","Holm","Hochberg","SidakSS","SidakSD","BH","BY","ABH","TSBH")
res2<-mt.rawp2adjp(rawp,procs)
```

---

mt.reject *Identity and number of rejected hypotheses*

Description

This function returns the identity and number of rejected hypotheses for several multiple testing procedures and different nominal Type I error rates.

Usage

```
mt.reject(adjp, alpha)
```

Arguments

- `adjp` A matrix of adjusted *p*-values, with rows corresponding to hypotheses and columns to multiple testing procedures. This matrix could be obtained from the function `mt.rawp2adjp`.
- `alpha` A vector of nominal Type I error rates.

Value

A list with components

- `r` A matrix containing the number of rejected hypotheses for several multiple testing procedures and different nominal Type I error rates. Rows correspond to Type I error rates and columns to multiple testing procedures.
- `which` A matrix of indicators for the rejection of individual hypotheses by different multiple testing procedures for a nominal Type I error rate `alpha[1]`. Rows correspond to hypotheses and columns to multiple testing procedures.

Author(s)

Sandrine Dudoit, `http://www.stat.berkeley.edu/~sandrine`,
Yongchao Ge, `<yongchao.ge@mssm.edu>`.

See Also

`mt.maxT`, `mt.minP`, `mt.rawp2adjp`, `golub`.

Examples

```
# Gene expression data from Golub et al. (1999)
# To reduce computation time and for illustrative purposes, we condider only
# the first 100 genes and use the default of B=10,000 permutations.
# In general, one would need a much larger number of permutations
# for microarray data.

data(golub)
smallgd<-golub[1:100,]
classlabel<-golub.cl

# Permutation unadjusted p-values and adjusted p-values for maxT procedure
res<-mt.maxT(smallgd,classlabel)
mt.reject(cbind(res$rawp,res$adjp),seq(0,1,0.1))$r
```

`mt.sample.teststat` *Permutation distribution of test statistics and raw (unadjusted) p-values*

Description

These functions provide tools to investigate the permutation distribution of test statistics, raw (unadjusted) p-values, and class labels.

Usage

```
mt.sample.teststat(V,classlabel,test="t",fixed.seed.sampling="y",B=10000,na=.mt.naNUM,nonpara="n")
mt.sample.rawp(V,classlabel,test="t",side="abs",fixed.seed.sampling="y",B=10000,na=.mt.naNUM,nonpar
mt.sample.label(classlabel,test="t",fixed.seed.sampling="y",B=10000)
```

Arguments

`V`	A numeric vector containing the data for one of the variables (genes).
`classlabel`	A vector of integers corresponding to observation (column) class labels. For k classes, the labels must be integers between 0 and $k-1$. For the `blockf` test option, observations may be divided into n/k blocks of k observations each. The observations are ordered by block, and within each block, they are labeled using the integers 0 to $k-1$.
`test`	A character string specifying the statistic to be used to test the null hypothesis of no association between the variables and the class labels. If `test="t"`, the tests are based on two-sample Welch t-statistics (unequal variances). If `test="t.equalvar"`, the tests are based on two-sample t-statistics with equal variance for the two samples. The square of the t-statistic is equal to an F-statistic for $k=2$. If `test="wilcoxon"`, the tests are based on standardized rank sum Wilcoxon

statistics.
If test="f", the tests are based on F-statistics.
If test="pair", the tests are based on paired t-statistics. The square of the paired t-statistic is equal to a block F-statistic for $k = 2$.
If test="blockf", the tests are based on F-statistics which adjust for block differences (cf. two-way analysis of variance).

side
A character string specifying the type of rejection region.
If side="abs", two-tailed tests, the null hypothesis is rejected for large absolute values of the test statistic.
If side="upper", one-tailed tests, the null hypothesis is rejected for large values of the test statistic.
If side="lower", one-tailed tests, the null hypothesis is rejected for small values of the test statistic.

fixed.seed.sampling
If fixed.seed.sampling="y", a fixed seed sampling procedure is used, which may double the computing time, but will not use extra memory to store the permutations. If fixed.seed.sampling="n", permutations will be stored in memory. For the blockf test, the option n was not implemented as it requires too much memory.

B
The number of permutations. For a complete enumeration, B should be 0 (zero) or any number not less than the total number of permutations.

na
Code for missing values (the default is .mt.naNUM=--93074815.62). Entries with missing values will be ignored in the computation, i.e., test statistics will be based on a smaller sample size. This feature has not yet fully implemented.

nonpara
If nonpara="y", nonparametric test statistics are computed based on ranked data.
If nonpara="n", the original data are used.

Value

For mt.sample.teststat, a vector containing B permutation test statistics.

For mt.sample.rawp, a vector containing B permutation unadjusted p-values.

For mt.sample.label, a matrix containing B sets of permuted class labels. Each row corresponds to one permutation.

Author(s)

Yongchao Ge, <yongchao.ge@mssm.edu>,
Sandrine Dudoit, http://www.stat.berkeley.edu/~sandrine.

See Also

mt.maxT, mt.minP, golub.

Examples

```
# Gene expression data from Golub et al. (1999)
data(golub)

mt.sample.label(golub.cl,B=10)

permt<-mt.sample.teststat(golub[1,],golub.cl,B=1000)
qqnorm(permt)
qqline(permt)

permt<-mt.sample.teststat(golub[50,],golub.cl,B=1000)
qqnorm(permt)
qqline(permt)

permp<-mt.sample.rawp(golub[1,],golub.cl,B=1000)
hist(permp)
```

`mt.teststat` *Computing test statistics for each row of a data frame*

Description

These functions provide a convenient way to compute test statistics, e.g., two-sample Welch t-statistics, Wilcoxon statistics, F-statistics, paired t-statistics, block F-statistics, for each row of a data frame.

Usage

```
mt.teststat(X,classlabel,test="t",na=.mt.naNUM,nonpara="n")
mt.teststat.num.denum(X,classlabel,test="t",na=.mt.naNUM,nonpara="n")
```

Arguments

`X` A data frame or matrix, with m rows corresponding to variables (hypotheses) and$n$ columns to observations. In the case of gene expression data, rows correspond to genes and columns to mRNA samples. The data can be read using `read.table`.

`classlabel` A vector of integers corresponding to observation (column) class labels. For k classes, the labels must be integers between 0 and $k-1$. For the `blockf` test option, observations may be divided into n/k blocks of k observations each. The observations are ordered by block, and within each block, they are labeled using the integers 0 to $k-1$.

`test` A character string specifying the statistic to be used to test the null hypothesis of no association between the variables and the class labels.
If `test="t"`, the tests are based on two-sample Welch t-statistics (unequal variances).
If `test="t.equalvar"`, the tests are based on two-sample t-statistics with equal

variance for the two samples. The square of the t-statistic is equal to an F-statistic for $k = 2$.
If `test="wilcoxon"`, the tests are based on standardized rank sum Wilcoxon statistics.
If `test="f"`, the tests are based on F-statistics.
If `test="pairt"`, the tests are based on paired t-statistics. The square of the paired t-statistic is equal to a block F-statistic for $k = 2$.
If `test="blockf"`, the tests are based on F-statistics which adjust for block differences (cf. two-way analysis of variance).

`na` Code for missing values (the default is `.mt.naNUM=--93074815.62`). Entries with missing values will be ignored in the computation, i.e., test statistics will be based on a smaller sample size. This feature has not yet fully implemented.

`nonpara` If `nonpara="y"`, nonparametric test statistics are computed based on ranked data.
If `nonpara="n"`, the original data are used.

Value

For `mt.teststat`, a vector of test statistics for each row (gene).

For `mt.teststat.num.denum`, a data frame with

`teststat.num` the numerator of the test statistics for each row, depending on the specific `test` option.

`teststat.denum` the denominator of the test statistics for each row, depending on the specific `test` option.

Author(s)

Yongchao Ge, <yongchao.ge@mssm.edu>,
Sandrine Dudoit, http://www.stat.berkeley.edu/~sandrine.

See Also

`mt.maxT`, `mt.minP`, `golub`.

Examples

```
# Gene expression data from Golub et al. (1999)
data(golub)

teststat<-mt.teststat(golub,golub.cl)
qqnorm(teststat)
qqline(teststat)

tmp<-mt.teststat.num.denum(golub,golub.cl,test="t")
num<-tmp$teststat.num
denum<-tmp$teststat.denum
plot(sqrt(denum),num)
```

```
tmp<-mt.teststat.num.denum(golub,golub.cl,test="f")
```

MTP *A function to perform resampling-based multiple hypothesis testing*

Description

A user-level function to perform multiple testing procedures (MTP). A variety of t- and F-tests, including robust versions of most tests, are implemented. Single-step and step-down minP and maxT methods are used to control the chosen type I error rate (FWER, gFWER, TPPFP, or FDR). Bootstrap and permutation null distributions are available. Additionally, for t-statistics, one may wish to sample from an appropriate multivariate normal distribution with mean zero and correlation matrix derived from the vector influence function. Arguments are provided for user control of output. Gene selection in microarray experiments is one application.

Usage

```
MTP(X, W = NULL, Y = NULL, Z = NULL, Z.incl = NULL, Z.test = NULL,
    na.rm = TRUE, test = "t.twosamp.unequalvar", robust = FALSE,
    standardize = TRUE, alternative = "two.sided", psi0 = 0,
    typeone = "fwer", k = 0, q = 0.1, fdr.method = "conservative",
    alpha = 0.05, smooth.null = FALSE, nulldist = "boot.cs",
    B = 1000, ic.quant.trans = FALSE, MVN.method = "mvrnorm",
    penalty = 1e-06, method = "ss.maxT", get.cr = FALSE, get.cutoff = FALSE,
    get.adjp = TRUE, keep.nulldist = TRUE, keep.rawdist = FALSE,
    seed = NULL, cluster = 1, type = NULL, dispatch = NULL, marg.null = NULL,
    marg.par = NULL, keep.margpar = TRUE, ncp = NULL, perm.mat = NULL,
    keep.index = FALSE, keep.label = FALSE)
```

Arguments

X — A matrix, data.frame or ExpressionSet containing the raw data. In the case of an ExpressionSet, `exprs(X)` is the data of interest and `pData(X)` may contain outcomes and covariates of interest. For most currently implemented tests (exception: tests involving correlation parameters), one hypothesis is tested for each row of the data.

W — A vector or matrix containing non-negative weights to be used in computing the test statistics. If a matrix, `W` must be the same dimension as `X` with one weight for each value in `X`. If a vector, `W` may contain one weight for each observation (i.e. column) of `X` or one weight for each variable (i.e. row) of `X`. In either case, the weights are duplicated appropriately. Weighted F-tests are not available. Default is 'NULL'.

Y — A vector, factor, or `Surv` object containing the outcome of interest. This may be class labels (F-tests and two sample t-tests) or a continuous or polycotomous dependent variable (linear regression based t-tests), or survival data (Cox proportional hazards based t-tests). For `block.f` and `f.twoway` tests, class labels must be ordered by block and within each block ordered by group. If `X` is an ExpressionSet, `Y` can be a character string referring to the column of `pData(X)` to use as outcome. Default is 'NULL'.

`Z` — A vector, factor, or matrix containing covariate data to be used in the regression (linear and Cox) models. Each variable should be in one column, so that `nrow(Z)=ncol(X)`. If `X` is an ExpressionSet, `Z` can be a character string referring to the column of `pData(X)` to use as covariates. The variables `Z.incl` and `Z.adj` allow one to specify which covariates to use in a particular test without modifying the input `Z`. Default is 'NULL'.

`Z.incl` — The indices of the columns of `Z` (i.e. which variables) to include in the model. These can be numbers or column names (if the columns are names). Default is 'NULL'.

`Z.test` — The index or names of the column of `Z` (i.e. which variable) to use to test for association with each row of `X` in a linear model. Only used for `test="lm.XvsZ"`, where it is necessary to specify which covariate's regression parameter is of interest. Default is 'NULL'.

`na.rm` — Logical indicating whether to remove observations with an NA. Default is 'TRUE'.

`test` — Character string specifying the test statistics to use, by default 't.twosamp.unequalvar'. See details (below) for a list of tests.

`robust` — Logical indicating whether to use the robust version of the chosen test, e.g. Wilcoxon singed rank test for robust one-sample t-test or `rlm` instead of `lm` in linear models. Default is 'FALSE'.

`standardize` — Logical indicating whether to use the standardized version of the test statistics (usual t-statistics are standardized). Default is 'TRUE'.

`alternative` — Character string indicating the alternative hypotheses, by default 'two.sided'. For one-sided tests, use 'less' or 'greater' for null hypotheses of 'greater than or equal' (i.e. alternative is 'less') and 'less than or equal', respectively.

`psi0` — The hypothesized null value, typically zero (default). Currently, this should be a single value, which is used for all hypotheses.

`typeone` — Character string indicating which type I error rate to control, by default family-wise error rate ('fwer'). Other options include generalized family-wise error rate ('gfwer'), with parameter `k` giving the allowed number of false positives, and tail probability of the proportion of false positives ('tppfp'), with parameter `q` giving the allowed proportion of false positives. The false discovery rate ('fdr') can also be controlled.

`k` — The allowed number of false positives for gFWER control. Default is 0 (FWER).

`q` — The allowed proportion of false positives for TPPFP control. Default is 0.1.

`fdr.method` — Character string indicating which FDR controlling method should be used when `typeone="fdr"`. The options are "conservative" (default) for the more conservative, general FDR controlling procedure and "restricted" for the method which requires more assumptions.

`alpha` — The target nominal type I error rate, which may be a vector of error rates. Default is 0.05.

smooth.null
: Indicator of whether to use a kernel density estimate for the tail of the null distributon for computing raw pvalues close to zero. Only used if 'rawp' would be zero without smoothing. Default is 'FALSE'.

nulldist
: Character string indicating which resampling method to use for estimating the joint test statistics null distribution, by default the non-parametric bootstrap with centering and scaling ('boot.cs'). The old default 'boot' will still compile and will correspond to 'boot.cs'. Other null distribution options include 'perm', 'boot.ctr', 'boot.qt', and 'ic', corresponding to the permutation distribution, centered-only bootstrap distribution, quantile-transformed bootstrap distribution, and influence curve multivariate normal joint null distribution, respectively. More details below.

B
: The number of bootstrap iterations (i.e. how many resampled data sets), the number of permutations (if `nulldist` is 'perm'), or the number of samples from the multivariate normal distribution (if `nulldist` is 'ic') Can be reduced to increase the speed of computation, at a cost to precision. Default is 1000.

ic.quant.trans
: If `nulldist='ic'`, a logical indicating whether or not a marginal quantile transformation using a t-distribution or user-supplied marginal distribution (stored in `perm.mat`) should be applied to the multivariate normal null distribution. Defaults for `marg.null` and `marg.par` exist, but can also be specified by the user (see below). Default is 'FALSE'.

MVN.method
: If `nulldist='ic'`, one of 'mvrnorm' or 'Cholesky' designating how correlated normal test statistics are to be generated. Selecting 'mvrnorm' uses the function of the same name found in the `MASS` library, whereas 'Cholesky' relies on a Cholesky decomposition. Default is 'mvrnorm'.

penalty
: If `nulldist='ic'` and `MVN.method='Cholesky'`, the value in `penalty` is added to all diagonal elements of the estimated test statistics correlation matrix to ensure that the matrix is positive definite and that internal calls to `'chol'` do not return an error. Default is 1e-6.

method
: The multiple testing procedure to use. Options are single-step maxT ('ss.maxT', default), single-step minP ('ss.minP'), step-down maxT ('sd.maxT'), and step-down minP ('sd.minP').

get.cr
: Logical indicating whether to compute confidence intervals for the estimates. Not available for F-tests. Default is 'FALSE'.

get.cutoff
: Logical indicating whether to compute thresholds for the test statistics. Default is 'FALSE'.

get.adjp
: Logical indicating whether to compute adjusted p-values. Default is 'TRUE'.

keep.nulldist
: Logical indicating whether to return the computed bootstrap or influence curve null distribution, by default 'TRUE'. Not available for `nulldist`='perm'. Note that this matrix can be quite large.

keep.rawdist
: Logical indicating whether to return the computed non-null (raw) bootstrap distribution, by default 'FALSE'. Not available when using `nulldist`='perm' or 'ic'. Note that this matrix can become quite large. If one wishes to use subsequent calls to `update` or `EBupdate` in which one updates choice of bootstrap null distribution, `keep.rawdist` must be TRUE. To save on memory, `update` only requires that one of `keep.nulldist` or `keep.rawdist` be 'TRUE'.

`seed` Integer or vector of integers to be used as argument to `set.seed` to set the seed for the random number generator for bootstrap resampling. This argument can be used to repeat exactly a test performed with a given seed. If the seed is specified via this argument, the same seed will be returned in the seed slot of the MTP object created. Else a random seed(s) will be generated, used and returned. Vector of integers used to specify seeds for each node in a cluster used to to generate a bootstrap null distribution.

`cluster` Integer for number of nodes to create or a cluster object created through the package snow. With `cluster=1`, bootstrap is implemented on single node. Supplying a cluster object results in the bootstrap being implemented in parallel on the provided nodes. This option is only available for the bootstrap procedure. With default value of 1, bootstrap is executed on single CPU.

`type` Interface system to use for computer cluster. See `snow` package for details.

`dispatch` The number or percentage of bootstrap iterations to dispatch at a time to each node of the cluster if a computer cluster is used. If dispatch is a percentage, `B*dispatch` must be an integer. If dispatch is an integer, then `B/dispatch` must be an integer. Default is 5 percent.

`marg.null` If `nulldist='boot.qt'`, the marginal null distribution to use for quantile transformation. Can be one of 'normal', 't', 'f' or 'perm'. Default is 'NULL', in which case the marginal null distribution is selected based on choice of test statistics. Defaults explained below. If 'perm', the user must supply a vector or matrix of test statistics corresponding to another marginal null distribution, perhaps one created externally by the user, and possibly referring to empirically derived *marginal permutation distributions*, although the statistics could represent any suitable choice of marginal null distribution.

`marg.par` If `nulldist='boot.qt'`, the parameters defining the marginal null distribution in `marg.null` to be used for quantile transformation. Default is 'NULL', in which case the values are selected based on choice of test statistics and other available parameters (e.g., sample size, number of groups, etc.). Defaults explained below. User can override defaults, in which case a matrix of marginal null distribution parameters can be accepted. Providing numeric (vector) values will apply the same null distribution defined by the parameter to all hypotheses, while providing a matrix of values allows the user to perform multiple testing using parameters which may vary with each hypothesis, as may be desired in common-quantile minP procedures. In this way, theoretical factors or factors affecting sample size or missingness may be assessed.

`keep.margpar` If `nulldist='boot.qt'`, a logical indicating whether the (internally created) matrix of marginal null distribution parameters should be returned. Default is 'TRUE'.

`ncp` If `nulldist='boot.qt'`, a value for a possible noncentrality parameter to be used during marginal quantile transformation. Default is 'NULL'.

`perm.mat` If `nulldist='boot.qt'` and `marg.null='perm'`, a matrix of user-supplied test statistics from a particular distribution to be used during marginal quantile transformation. The statistics may represent empirically derived marginal permutation values, may be theoretical values, or may represent a sample from some other suitable choice of marginal null distribution.

`keep.index` If `nulldist='ic'` and `test='t.cor'` or `test='z.cor'`, the index returned is a matrix with the indices of the first and second variables considered for pairwise correlations. If there are p hypotheses, this arguments returns `t(combn(p,2))`. For all other choices of test statistic, the index is not returned, as they correspond to the original order of the hypotheses in `X`.

`keep.label` Default is 'FALSE'. A logical indicating whether or not the label in `Y` should be returned as a slot in the resulting MTP object. Typically not necessary, although useful if one is using `update` and wants to use marginal null distribution defaults with `nulldist='boot.qt'` (e.g., with F-tests).

Details

A multiple testing procedure (MTP) is defined by choices of test statistics, type I error rate, null distribution and method for error rate control. Each component is described here. For two-sample t-tests, the group with the smaller-valued label is substracted from the group with the larger-valued label. That is, differences in means are calculated as "mean of group 2 - mean of group 1" or "mean of group B - mean of group A". For paired t-tests, the arrangement of group indices does not matter, as long as the columns are arranged in the same corresponding order between groups. For example, if group 1 is coded as 0, and group 2 is coded as 1, for 3 pairs of data, it does not matter if the label `Y` is coded as "0,0,0,1,1,1", "1,1,1,0,0,0" "0,1,0,1,0,1" or "1,0,1,0,1,0", the paired differences between groups will be calculated as "group 2 - group 1". See references for more detail.

Test statistics are determined by the values of `test`:

t.onesamp: one-sample t-statistic for tests of means;

t.twosamp.equalvar: equal variance two-sample t-statistic for tests of differences in means (two-sample t-statistic);

t.twosamp.unequalvar: unequal variance two-sample t-statistic for tests of differences in means (two-sample Welch t-statistic);

t.pair: two-sample paired t-statistic for tests of differences in means;

f: multi-sample F-statistic for tests of equality of population means (assumes constant variance across groups, but not normality);

f.block: multi-sample F-statistic for tests of equality of population means in a block design (assumes constant variance across groups, but not normality). This test is not available with the bootstrap null distribution;

f.twoway: multi-sample F-statistic for tests of equality of population means in a block design (assumes constant variance across groups, but not normality). Differs from `f.block` in requiring multiple observations per group*block combintation. This test uses the means of each group*block combination as response variable and test for group main effects assuming a randomized block design;

lm.XvsZ: t-statistic for tests of regression coefficients for variable `Z.test` in linear models, each with a row of X as outcome, possibly adjusted by covariates `Z.incl` from the matrix `Z` (in the case of no covariates, one recovers the one-sample t-statistic, `t.onesamp`);

lm.YvsXZ: t-statistic for tests of regression coefficients in linear models, with outcome Y and each row of X as covariate of interest, with possibly other covariates `Z.incl` from the matrix `Z`;

coxph.YvsXZ: t-statistic for tests of regression coefficients in Cox proportional hazards survival models, with outcome Y and each row of X as covariate of interest, with possibly other covariates `Z.incl` from the matrix `Z`.

t.cor t-statistics for tests of pairwise correlation parameters for all variables in X. Note that the number of hypotheses can become quite large very fast. This test is only available with the influence curve null distribution.

z.cor Fisher's z-statistics for tests of pairwise correlation parameters for all variables in X. Note that the number of hypotheses can become quite large very fast. This test is only available with the influence curve null distribution.

When `robust=TRUE`, non-parametric versions of each test are performed. For the linear models, this means `rlm` is used instead of `lm`. There is not currently a robust version of `test=coxph.YvsXZ`. For the t- and F-tests, data values are simply replaced by their ranks. This is equivalent to performing the following familiar named rank-based tests. The conversion after each test is the formula to convert from the MTP test to the statistic reported by the listed R function (where num is the numerator of the MTP test statistics, n is total sample size, nk is group k sample size, K is total number of groups or treatments, and rk are the ranks in group k).

t.onesamp or t.pair: Wilcoxon signed rank, `wilcox.test` with `y=NULL` or `paired=TRUE`, conversion: num/n

t.twosamp.equalvar: Wilcoxon rank sum or Mann-Whitney, `wilcox.test`, conversion: n2*(num+mean(r1)) - n2*(n2+1)/2

f: Kruskal-Wallis rank sum, `kruskal.test`, conversion: num*12/(n*(n-1))

f.block: Friedman rank sum, `friedman.test`, conversion: num*12/(K*(K+1))

f.twoway: Friedman rank sum, `friedman.test`, conversion: num*12/(K*(K+1))

The implemented MTPs are based on control of the family-wise error rate, defined as the probability of any false positives. Let Vn denote the (unobserved) number of false positives. Then, control of FWER at level alpha means that Pr(Vn>0)<=alpha. The set of rejected hypotheses under a FWER controlling procedure can be augmented to increase the number of rejections, while controlling other error rates. The generalized family-wise error rate is defined as Pr(Vn>k)<=alpha, and it is clear that one can simply take an FWER controlling procedure, reject k more hypotheses and have control of gFWER at level alpha. The tail probability of the proportion of false positives depends on both the number of false postives (Vn) and the number of rejections (Rn). Control of TPPFP at level alpha means Pr(Vn/Rn>q)<=alpha, for some proportion q. Control of the false discovery rate refers to the expected proportion of false positives (rather than a tail probability). Control of FDR at level alpha means E(Vn/Rn)<=alpha.

In practice, one must choose a method for estimating the test statistics null distribution. We have implemented several versions of an ordinary non-parametric bootstrap estimator and a permutation estimator (which makes sense in certain settings, see references). The non-parametric bootstrap estimator (default) provides asymptotic control of the type I error rate for any data generating distribution, whereas the permutation estimator requires the subset pivotality assumption. One draw back of both methods is the discreteness of the estimated null distribution when the sample size is small. Furthermore, when the sample size is small enough, it is possible that ties will lead to a very small variance estimate. Using `standardize=FALSE` allows one to avoid these unusually small test statistic denominators. Parametric bootstrap estimators are another option (not yet implemented). For asymptotically linear estimators, such as those commonly probed using t-statistics, another choice of null distribution is provided when sampling from a multivariate normal distribution with mean zero and correlation matrix derived from the vector influence function. Sampling from a multivariate normal may alleviate the discreteness of the bootstrap and permutation distributions, although accuracy in estimation of the test statistics correlation matrix will be of course also affected by sample size.

For the nonparametric bootstrap distribution with marginal null quantile transformation, the following defaults for `marg.null` and `marg.par` are available based on choice of test statistics, sample size 'n', and various other parameters:

t.onesamp: t-distribution with df=n-1;

t.twosamp.equalvar: t-distribution with df=n-2;

t.twosamp.unequalvar: N(0,1);

t.pair: t-distribution with df=n-1, where n is the number of unique samples, i.e., the number of observed differences between paired samples;

f: F-distribution with df1=k-1, df2=n-k, for k groups;

f.block: NA. Only available with permutation distribution;

f.twoway: F-distribution with df1=k-1,df2=n-k*l, for k groups and l blocks;

lm.XvsZ: N(0,1);

lm.YvsXZ: N(0,1);

coxph.YvsXZ: N(0,1);

t.cor t-distribution with df=n-2;

z.cor N(0,1).

The above defaults, however, can be overridden by manually setting values of `marg.null` and `marg.par`. In the case of `nulldist='ic'`, and `ic.quant.trans=TRUE`, the defaults are the same as above except that 'lm.XvsZ' and 'lm.YvsXZ' are replaced with t-distributions with df=n-p.

Given observed test statistics, a type I error rate (with nominal level), and a test statistics null distribution, MTPs provide adjusted p-values, cutoffs for test statistics, and possibly confidence regions for estimates. Four methods are implemented, based on minima of p-values and maxima of test statistics. Only the step down methods are currently available with the permutation null distribution.

Computation times using a bootstrap null distribution are slower when weights are used for one and two-sample tests. Computation times when using a bootstrap null distribution also are slower for the tests `lmXvsZ`, `lmYvsXZ`, `coxph.YvsXZ`.

To execute the bootstrap on a computer cluster, a cluster object generated with `makeCluster` in the package `snow` may be used as the argument for cluster. Alternatively, the number of nodes to use in the computer cluster can be used as the argument to cluster. In this case, `type` must be specified and a cluster will be created. In both cases, `Biobase` and `multtest` will be loaded onto each cluster node if these libraries are located in a directory in the standard search path. If these libraries are in a non-standard location, it is necessary to first create the cluster, load `Biobase` and `multtest` on each node and then to use the cluster object as the argument to cluster. See documentation for `snow` package for additional information on creating and using a cluster.

Finally, note that the old argument `csnull` is now DEPRECATED as of `multtest` v. 2.0.0 given the expanded null distribution options described above. Previously, this argument was an indicator of whether the bootstrap estimated test statistics distribution should be centered and scaled (to produce a null distribution) or not. If csnull=FALSE, the (raw) non-null bootstrap estimated test statistics distribution was returned. If the non-null bootstrap distribution should be returned, this object is now stored in the 'rawdist' slot when keep.rawdist=TRUE in the original MTP function call.

Value

An object of class MTP, with the following slots:

statistic
: Object of class numeric, observed test statistics for each hypothesis, specified by the values of the MTP arguments test, robust, standardize, and psi0.

estimate
: For the test of single-parameter null hypotheses using t-statistics (i.e., not the F-tests), the numeric vector of estimated parameters corresponding to each hypothesis, e.g. means, differences in means, regression parameters.

sampsize
: Object of class numeric, number of columns (i.e. observations) in the input data set.

rawp
: Object of class numeric, unadjusted, marginal p-values for each hypothesis.

adjp
: Object of class numeric, adjusted (for multiple testing) p-values for each hypothesis (computed only if the get.adjp argument is TRUE).

conf.reg
: For the test of single-parameter null hypotheses using t-statistics (i.e., not the F-tests), the numeric array of lower and upper simultaneous confidence limits for the parameter vector, for each value of the nominal Type I error rate alpha (computed only if the get.cr argument is TRUE).

cutoff
: The numeric matrix of cut-offs for the vector of test statistics for each value of the nominal Type I error rate alpha (computed only if the get.cutoff argument is TRUE).

reject
: Object of class 'matrix', rejection indicators (TRUE for a rejected null hypothesis), for each value of the nominal Type I error rate alpha.

rawdist
: The numeric matrix for the estimated nonparametric non-null test statistics distribution (returned only if keep.rawdist=TRUE and if nulldist is one of 'boot.ctr', 'boot.cs', or 'boot.qt'). This slot must not be empty if one wishes to call update to change choice of bootstrap-based null distribution.

nulldist
: The numeric matrix for the estimated test statistics null distribution (returned only if keep.nulldist=TRUE); option not currently available for permutation null distribution, i.e., nulldist='perm'). By default (i.e., for nulldist='boot.cs'), the entries of nulldist are the null value shifted and scaled bootstrap test statistics, with one null test statistic value for each hypothesis (rows) and bootstrap iteration (columns).

nulldist.type
: Character value describing which choice of null distribution was used to generate the MTP results. Takes on one of the values of the original nulldist argument in the call to MTP, i.e., 'boot.cs', 'boot.ctr', 'boot.qt', 'ic', or 'perm'.

marg.null
: If nulldist='boot.qt', a character value returning which choice of marginal null distribution was used by the MTP. Can be used to check default values or to ensure manual settings were correctly applied.

marg.par
: If `nulldist='boot.qt'`, a numeric matrix returning the parameters of the marginal null distribution(s) used by the MTP. Can be used to check default values or to ensure manual settings were correctly applied.

call
: Object of class `call`, the call to the MTP function.

seed
: An integer or vector for specifying the state of the random number generator used to create the resampled datasets. The seed can be reused for reproducibility in a repeat call to `MTP`. This argument is currently used only for the bootstrap null distribution (i.e., for `nulldist="boot.xx"`). See `?set.seed` for details.

Note

Thank you to Peter Dimitrov for suggestions about the code.

Author(s)

Katherine S. Pollard and Houston N. Gilbert with design contributions from Sandra Taylor, Sandrine Dudoit and Mark J. van der Laan.

References

M.J. van der Laan, S. Dudoit, K.S. Pollard (2004), Augmentation Procedures for Control of the Generalized Family-Wise Error Rate and Tail Probabilities for the Proportion of False Positives, Statistical Applications in Genetics and Molecular Biology, 3(1). http://www.bepress.com/sagmb/vol3/iss1/art15/

M.J. van der Laan, S. Dudoit, K.S. Pollard (2004), Multiple Testing. Part II. Step-Down Procedures for Control of the Family-Wise Error Rate, Statistical Applications in Genetics and Molecular Biology, 3(1). http://www.bepress.com/sagmb/vol3/iss1/art14/

S. Dudoit, M.J. van der Laan, K.S. Pollard (2004), Multiple Testing. Part I. Single-Step Procedures for Control of General Type I Error Rates, Statistical Applications in Genetics and Molecular Biology, 3(1). http://www.bepress.com/sagmb/vol3/iss1/art13/

K.S. Pollard and Mark J. van der Laan, "Resampling-based Multiple Testing: Asymptotic Control of Type I Error and Applications to Gene Expression Data" (June 24, 2003). U.C. Berkeley Division of Biostatistics Working Paper Series. Working Paper 121. http://www.bepress.com/ucbbiostat/paper121

M.J. van der Laan and A.E. Hubbard (2006), Quantile-function Based Null Distributions in Resampling Based Multiple Testing, Statistical Applications in Genetics and Molecular Biology, 5(1). http://www.bepress.com/sagmb/vol5/iss1/art14/

S. Dudoit and M.J. van der Laan. Multiple Testing Procedures and Applications to Genomics. Springer Series in Statistics. Springer, New York, 2008.

See Also

`EBMTP`, `MTP-class`, `MTP-methods`, `mt.minP`, `mt.maxT`, `ss.maxT`, `fwer2gfwer`

Examples

```
#data
set.seed(99)
data<-matrix(rnorm(90),nr=9)
group<-c(rep(1,5),rep(0,5))

#fwer control with centered and scaled bootstrap null distribution
#(B=100 for speed)
m1<-MTP(X=data,Y=group,alternative="less",B=100,method="sd.minP")
print(m1)
summary(m1)
par(mfrow=c(2,2))
plot(m1,top=9)

#fwer control with quantile transformed bootstrap null distribution
#default settings = N(0,1) marginal null distribution
m2<-MTP(X=data,Y=group,alternative="less",B=100,method="sd.minP",
 nulldist="boot.qt",keep.rawdist=TRUE)

#fwer control with quantile transformed bootstrap null distribution
#marginal null distribution and df parameters manually set,
#first all equal, then varying with hypothesis
m3<-update(m2,marg.null="t",marg.par=10)
mps<-matrix(c(rep(9,5),rep(10,5)),nr=10,nc=1)
m4<-update(m2,marg.null="t",marg.par=mps)

m1@nulldist.type
m2@nulldist.type
m2@marg.null
m2@marg.par
m3@nulldist.type
m3@marg.null
m3@marg.par
m4@nulldist.type
m4@marg.null
m4@marg.par
```

MTP-class *Class "MTP", classes and methods for multiple testing procedure output*

Description

An object of class MTP is the output of a particular multiple testing procedure, for example, generated by the MTP function. It has slots for the various data used to make multiple testing decisions, such as adjusted p-values and confidence regions.

Objects from the Class

Objects can be created by calls of the form
new('MTP',
statistic = ..., object of class numeric
estimate = ..., object of class numeric
sampsize = ..., object of class numeric
rawp = ..., object of class numeric
adjp = ..., object of class numeric
conf.reg = ..., object of class array
cutoff = ..., object of class matrix
reject = ..., object of class matrix
rawdist = ..., object of class matrix
nulldist = ..., object of class matrix
nulldist.type = ..., object of class character
marg.null = ..., object of class character
marg.par = ..., object of class matrix
label = ..., object of class numeric
index = ..., object of class matrix
call = ..., object of class call
seed = ..., object of class integer
)

Slots

`statistic` Object of class `numeric`, observed test statistics for each hypothesis, specified by the values of the `MTP` arguments `test`, `robust`, `standardize`, and `psi0`.

`estimate` For the test of single-parameter null hypotheses using t-statistics (i.e., not the F-tests), the numeric vector of estimated parameters corresponding to each hypothesis, e.g. means, differences in means, regression parameters.

`sampsize` Object of class `numeric`, number of columns (i.e. observations) in the input data set.

`rawp` Object of class `numeric`, unadjusted, marginal p-values for each hypothesis.

`adjp` Object of class `numeric`, adjusted (for multiple testing) p-values for each hypothesis (computed only if the `get.adjp` argument is TRUE).

`conf.reg` For the test of single-parameter null hypotheses using t-statistics (i.e., not the F-tests), the numeric array of lower and upper simultaneous confidence limits for the parameter vector, for each value of the nominal Type I error rate `alpha` (computed only if the `get.cr` argument is TRUE).

`cutoff` The numeric matrix of cut-offs for the vector of test statistics for each value of the nominal Type I error rate `alpha` (computed only if the `get.cutoff` argument is TRUE).

`reject` Object of class `'matrix'`, rejection indicators (TRUE for a rejected null hypothesis), for each value of the nominal Type I error rate `alpha`.

`rawdist` The numeric matrix for the estimated nonparametric non-null test statistics distribution (returned only if `keep.rawdist=TRUE` and if `nulldist` is one of 'boot.ctr', 'boot.cs', or 'boot.qt'). This slot must not be empty if one wishes to call `update` to change choice of bootstrap-based null distribution.

`nulldist` The numeric matrix for the estimated test statistics null distribution (returned only if `keep.nulldist=TRUE`); option not currently available for permutation null distribution, i.e., `nulldist='perm'`). By default (i.e., for `nulldist='boot.cs'`), the entries of `nulldist` are the null value shifted and scaled bootstrap test statistics, with one null test statistic value for each hypothesis (rows) and bootstrap iteration (columns).

`nulldist.type` Character value describing which choice of null distribution was used to generate the MTP results. Takes on one of the values of the original `nulldist` argument in the call to MTP, i.e., 'boot.cs', 'boot.ctr', 'boot.qt', 'ic', or 'perm'.

`marg.null` If `nulldist='boot.qt'`, a character value returning which choice of marginal null distribution was used by the MTP. Can be used to check default values or to ensure manual settings were correctly applied.

`marg.par` If `nulldist='boot.qt'`, a numeric matrix returning the parameters of the marginal null distribution(s) used by the MTP. Can be used to check default values or to ensure manual settings were correctly applied.

`label` If `keep.label=TRUE`, a vector storing the values used in the argument `Y`. Storing this object is particularly important when one wishes to update MTP objects with F-statistics using default `marg.null` and `marg.par` settings when `nulldist='boot.qt'`.

`index` For tests of correlation parameters a matrix corresponding to `t(combn(p,2))`, where `p` is the number of variables in `X`. This matrix gives the indices of the variables considered in each pairwise correlation. For all other tests, this slot is empty, as the indices are in the same order as the rows of `X`.

`call` Object of class `call`, the call to the MTP function.

`seed` An integer or vector for specifying the state of the random number generator used to create the resampled datasets. The seed can be reused for reproducibility in a repeat call to `MTP`. This argument is currently used only for the bootstrap null distribution (i.e., for `nulldist="boot.xx"`). See `?set.seed` for details.

Methods

`signature(x = "MTP")`

[: Subsetting method for `MTP` class, which operates selectively on each slot of an `MTP` instance to retain only the data related to the specified hypotheses.

as.list : Converts an object of class `MTP` to an object of class `list`, with an entry for each slot.

plot : plot methods for `MTP` class, produces the following graphical summaries of the results of a MTP. The type of display may be specified via the `which` argument.

1. Scatterplot of number of rejected hypotheses vs. nominal Type I error rate.

2. Plot of ordered adjusted p-values; can be viewed as a plot of Type I error rate vs. number of rejected hypotheses.

3. Scatterplot of adjusted p-values vs. test statistics (also known as "volcano plot").

4. Plot of unordered adjusted p-values.

5. Plot of confidence regions for user-specified parameters, by default the 10 parameters corresponding to the smallest adjusted p-values (argument `top`).

6. Plot of test statistics and corresponding cut-offs (for each value of `alpha`) for user-specified hypotheses, by default the 10 hypotheses corresponding to the smallest adjusted p-values (argument `top`).

The argument `logscale` (by default equal to FALSE) allows one to use the negative decimal logarithms of the adjusted p-values in the second, third, and fourth graphical displays. The arguments `caption` and `sub.caption` allow one to change the titles and subtitles for each of the plots (default subtitle is the MTP function call). Note that some of these plots are implemented in the older function `mt.plot`.

print : print method for `MTP` class, returns a description of an object of class `MTP`, including sample size, number of tested hypotheses, type of test performed (value of argument `test`), Type I error rate (value of argument `typeone`), nominal level of the test (value of argument `alpha`), name of the MTP (value of argument `method`), call to the function `MTP`.

In addition, this method produces a table with the class, mode, length, and dimension of each slot of the `MTP` instance.

summary : summary method for `MTP` class, provides numerical summaries of the results of a MTP and returns a list with the following three components.

1. rejections: A data.frame with the number(s) of rejected hypotheses for the nominal Type I error rate(s) specified by the `alpha` argument of the function `MTP`. (NULL values are returned if all three arguments `get.cr`, `get.cutoff`, and `get.adjp` are FALSE).

2. index: A numeric vector of indices for ordering the hypotheses according to first `adjp`, then `rawp`, and finally the absolute value of `statistic` (not printed in the summary).

3. summaries: When applicable (i.e., when the corresponding quantities are returned by `MTP`), a table with six number summaries of the distributions of the adjusted p-values, unadjusted p-values, test statistics, and parameter estimates.

update : update method for `MTP` class, provides a mechanism to re-run the MTP with different choices of the following arguments - nulldist, alternative, typeone, k, q, fdr.method, alpha, smooth.null, method, get.cr, get.cutoff, get.adjp, keep.nulldist, keep.rawdist, keep.margpar. When evaluate is 'TRUE', a new object of class MTP is returned. Else, the updated call is returned. The `MTP` object passed to the update method must have either a non-empty `rawdist` slot or a non-empty `nulldist` slot (i.e., must have been called with either 'keep.rawdist=TRUE' or 'keep.nulldist=TRUE').

To save on memory, if one knows ahead of time that one will want to compare different choices of bootstrap-based null distribution, then it is both necessary and sufficient to specify 'keep.rawdist=TRUE', as there is no other means of moving between null distributions other than through the non-transformed non-parametric bootstrap distribution. In this case, 'keep.nulldist=FALSE' may be used. Specifically, if an object of class `MTP` contains a non-empty `rawdist` slot and an empty `nulldist` slot, then a new null distribution will be generated according to the values of the nulldist= argument in the original call to MTP or any additional specifications in the call to update. On the other hand, if one knows that one wishes to only update an MTP object in ways which do not involve choice of null distribution, then 'keep.nulldist=TRUE' will suffice and 'keep.rawdist' can be set to FALSE (default settings). The original null distribution object will then be used for all subsequent calls to update.

N.B.: Note that keep.rawdist=TRUE is only available for the bootstrap-based resampling methods. The non-null distribution does not exist for the permutation or influence curve multivariate normal null distributions.

mtp2ebmtp : coersion method for converting objects of class MTP to objects of class EBMTP. Slots common to both objects are taken from the object of class MTP and used to create a new object of class EBMTP. Once an object of class EBMTP is created, one may use the method EBupdate to perform resampling-based empirical Bayes multiple testing without the need for repeated resampling.

Author(s)

Katherine S. Pollard and Houston N. Gilbert with design contributions from Sandrine Dudoit and Mark J. van der Laan.

References

M.J. van der Laan, S. Dudoit, K.S. Pollard (2004), Augmentation Procedures for Control of the Generalized Family-Wise Error Rate and Tail Probabilities for the Proportion of False Positives, Statistical Applications in Genetics and Molecular Biology, 3(1). http://www.bepress.com/sagmb/vol3/iss1/art15/

M.J. van der Laan, S. Dudoit, K.S. Pollard (2004), Multiple Testing. Part II. Step-Down Procedures for Control of the Family-Wise Error Rate, Statistical Applications in Genetics and Molecular Biology, 3(1). http://www.bepress.com/sagmb/vol3/iss1/art14/

S. Dudoit, M.J. van der Laan, K.S. Pollard (2004), Multiple Testing. Part I. Single-Step Procedures for Control of General Type I Error Rates, Statistical Applications in Genetics and Molecular Biology, 3(1). http://www.bepress.com/sagmb/vol3/iss1/art13/

Katherine S. Pollard and Mark J. van der Laan, "Resampling-based Multiple Testing: Asymptotic Control of Type I Error and Applications to Gene Expression Data" (June 24, 2003). U.C. Berkeley Division of Biostatistics Working Paper Series. Working Paper 121. http://www.bepress.com/ucbbiostat/paper121

M.J. van der Laan and A.E. Hubbard (2006), Quantile-function Based Null Distributions in Resampling Based Multiple Testing, Statistical Applications in Genetics and Molecular Biology, 5(1). http://www.bepress.com/sagmb/vol5/iss1/art14/

S. Dudoit and M.J. van der Laan. Multiple Testing Procedures and Applications to Genomics. Springer Series in Statistics. Springer, New York, 2008.

See Also

MTP, MTP-methods, EBMTP, EBMTP-methods, [-methods, as.list-methods, print-methods, plot-methods, summary-methods, mtp2ebmtp, ebmtp2mtp

Examples

```
## See MTP function: ? MTP
```

MTP-methods *Methods for MTP and EBMTP objects in Package 'multtest'*

Description

Summary, printing, plotting, subsetting, updating, as.list and class conversion methods were defined for the MTP and EBMTP classes. These methods provide visual and numeric summaries of the results of a multiple testing procedure (MTP) and allow one to perform some basic manipulations of objects class MTP or EBMTP.

Several of the methods with the same name will work on objects of their respective class. One exception to this rule is the difference between update and EBupdate (described below). Because of the differences in the testing procedures, separately named methods were chosen to clearly delineate which method was being applied to which type of object.

Methods

[: Subsetting method for MTP and EBMTP classes, which operates selectively on each slot of an MTP or EBMTP instance to retain only the data related to the specified hypotheses.

as.list : Converts an object of class MTP or EBMTP to an object of class list, with an entry for each slot.

plot : plot methods for MTP and EBMTP classes, produces the following graphical summaries of the results of a MTP. The type of display may be specified via the which argument.

1. Scatterplot of number of rejected hypotheses vs. nominal Type I error rate.

2. Plot of ordered adjusted p-values; can be viewed as a plot of Type I error rate vs. number of rejected hypotheses.

3. Scatterplot of adjusted p-values vs. test statistics (also known as volcano plot).

4. Plot of unordered adjusted p-values.

Only for objects of class MTP:

5. Plot of confidence regions for user-specified parameters, by default the 10 parameters corresponding to the smallest adjusted p-values (argument top).

6. Plot of test statistics and corresponding cut-offs (for each value of alpha) for user-specified hypotheses, by default the 10 hypotheses corresponding to the smallest adjusted p-values (argument top).

Plots (5) and (6) are not available for objects of class EBMTP because the function EBMTP returns only adjusted p-values and not confidence regions of cut-offs. The argument logscale (by default equal to FALSE) allows one to use the negative decimal logarithms of the adjusted p-values in the second, third, and fourth graphical displays. The arguments caption and sub.caption allow one to change the titles and subtitles for each of the plots (default subtitle is the MTP function call). Note that some of these plots are implemented in the older function mt.plot.

print : print method for MTP and EBMTP classes, returns a description of an object of either class, including sample size, number of tested hypotheses, type of test performed (value of argument test), Type I error rate (value of argument typeone), nominal level of the test (value of argument alpha), name of the MTP (value of argument method), call to the function MTP or EBMTP.

In addition, this method produces a table with the class, mode, length, and dimension of each slot of the MTP or EBMTP instance.

summary : summary method for MTP and EBMTP classes, provides numerical summaries of the results of a MTP and returns a list with the following three components.

1. rejections: A data.frame with the number(s) of rejected hypotheses for the nominal Type I error rate(s) specified by the alpha argument of the function MTP or EBMTP. (For objects of class MTP, NULL values are returned if all three arguments get.cr, get.cutoff, and get.adjp are FALSE).

2. index: A numeric vector of indices for ordering the hypotheses according to first adjp, then rawp, and finally the absolute value of statistic (not printed in the summary).

3. summaries: When applicable (i.e., when the corresponding quantities are returned by MTP or EBMTP), a table with six number summaries of the distributions of the adjusted p-values, unadjusted p-values, test statistics, and parameter estimates.

update : update methods for MTP class, respectively, provides a mechanism to re-run the MTP with different choices of the following arguments - nulldist, alternative, typeone, k, q, fdr.method, alpha, smooth.null, method, get.cr, get.cutoff, get.adjp, keep.nulldist, keep.rawdist, keep.margpar. When evaluate is 'TRUE', a new object of class MTP is returned. Else, the updated call is returned. The MTP object passed to the update method must have either a non-empty rawdist slot or a non-empty nulldist slot (i.e., must have been called with either 'keep.rawdist=TRUE' or 'keep.nulldist=TRUE').

EBupdate : update method for EBMTP class, provides a mechanism to re-run the MTP with different choices of the following arguments - nulldist, alternative, typeone, k, q, alpha, smooth.null, bw, kernel, prior, keep.nulldist, keep.rawdist, keep.falsepos, keep.truepos, keep.errormat, keep.margpar. When evaluate is 'TRUE', a new object of class EBMTP is returned. Else, the updated call is returned. The EBMTP object passed to the update method must have either a non-empty rawdist slot or a non-empty nulldist slot (i.e., must have been called with either 'keep.rawdist=TRUE' or 'keep.nulldist=TRUE').

Additionally, when calling EBupdate for any Type I error rate other than FWER, the typeone argument must be specified (even if the original object did not control FWER). For example, typeone="fdr", would always have to be specified, even if the original object also controlled the FDR. In other words, for all function arguments, it is safest to always assume that you are updating from the `EBMTP` default function settings, regardless of the original call to the `EBMTP` function. Currently, the main advantage of the `EBupdate` method is that it prevents the need for repeated estimation of the test statistics null distribution.

To save on memory, if one knows ahead of time that one will want to compare different choices of bootstrap-based null distribution, then it is both necessary and sufficient to specify 'keep.rawdist=TRUE', as there is no other means of moving between null distributions other than through the non-transformed non-parametric bootstrap distribution. In this case, 'keep.nulldist=FALSE' may be used. Specifically, if an object of class `MTP` or `EBMTP` contains a non-empty `rawdist` slot and an empty `nulldist` slot, then a new null distribution will be generated according to the values of the `nulldist=` argument in the original call to `(EB)MTP` or any additional specifications in the call to `(EB)update`. On the other hand, if one knows that one wishes to only update an `(EB)MTP` object in ways which do not involve choice of bootstrap null distribution, then 'keep.nulldist=TRUE' will suffice and 'keep.rawdist' can be set to `FALSE` (default settings). The original null distribution object will then be used for all subsequent calls to `update`.

N.B.: Note that `keep.rawdist=TRUE` is only available for the bootstrap-based resampling methods. The non-null distribution does not exist for the permutation or influence curve multivariate normal null distributions.

mtp2ebmtp : coersion method for converting objects of class `MTP` to objects of class `EBMTP`. Slots common to both objects are taken from the object of class `MTP` and used to create a new object of class `EBMTP`. Once an object of class `EBMTP` is created, one may use the method `EBupdate` to perform resampling-based empirical Bayes multiple testing without the need for repeated resampling.

ebmtp2mtp : coersion method for converting objects of class `EBMTP` to objects of class `MTP`. Slots common to both objects are taken from the object of class `EBMTP` and used to create a new object of class `MTP`. Once an object of class `MTP` is created, one may use the method `update` to perform resampling-based multiple testing (as would have been done with calls to `MTP`) without the need for repeated resampling.

Author(s)

Katherine S. Pollard and Houston N. Gilbert with design contributions from Sandrine Dudoit and Mark J. van der Laan.

`multtest-internal` *Internal multtest functions and variables*

Description

Internal multtest functions and variables

Usage

```
.mt.BLIM
.mt.RandSeed
.mt.naNUM
mt.number2na(x,na)
mt.na2number(x,na)
mt.getmaxB(classlabel,test,B, verbose)
mt.transformL(classlabel,test)
mt.transformV(V,classlabel,test,na,nonpara)
mt.transformX(X,classlabel,test,na,nonpara)
mt.checkothers(side="abs",fixed.seed.sampling="y", B=10000,
na=.mt.naNUM, nonpara="n")
mt.checkX(X,classlabel,test)
mt.checkV(V,classlabel,test)
mt.checkclasslabel(classlabel,test)
mt.niceres<-function(res,X,index)
mt.legend(x, y = NULL, legend, fill = NULL, col = "black", lty,
    lwd, pch, angle = 45, density = NULL, bty = "o", bg = par("bg"),
    pt.bg = NA, cex = 1, pt.cex = cex, pt.lwd = lwd, xjust = 0,
    yjust = 1, x.intersp = 1, y.intersp = 1, adj = c(0, 0.5),
    text.width = NULL, text.col = par("col"), merge = do.lines &&
        has.pch, trace = FALSE, plot = TRUE, ncol = 1, horiz = FALSE,...)
corr.Tn(X, test, alternative, use = "pairwise")
diffs.1.N(vec1, vec2, e1, e2, e21, e22, e12)
IC.Cor.NA(IC, W, N, M, output)
IC.CorXW.NA(X, W, N, M, output)
insert.NA(orig.NA, res.vec)
marg.samp(marg.null, marg.par, m, B, ncp)
```

Details

These are not to be called directly by the user.

---

ss.maxT *Procedures to perform multiple testing*

Description

Given observed test statistics, a test statistics null distribution, and alternetive hyptheses, these multiple testing procedures provide family-wise error rate (FWER) adjusted p-values, cutoffs for test statistics, and possibly confidence regions for estimates. Four methods are implemented, based on minima of p-values and maxima of test statistics.

Usage

```
ss.maxT(null, obs, alternative, get.cutoff, get.cr,
get.adjp, alpha = 0.05)
```

```
ss.minP(null, obs, rawp, alternative, get.cutoff, get.cr,
get.adjp, alpha=0.05)

sd.maxT(null, obs, alternative, get.cutoff, get.cr,
get.adjp, alpha = 0.05)

sd.minP(null, obs, rawp, alternative, get.cutoff, get.cr,
get.adjp, alpha=0.05)
```

Arguments

`null`	A matrix containing the test statistics null distribution, e.g. the output of `boot.resample`.
`obs`	A vector of observed test statistics, e.g. the output of a test statistics closure such as `meanX`. These are stored as a matrix with numerator (possibly absolute value or negative, depending on the value of alternative) in the first row, denominator in the second row, and a 1 or -1 in the third row (depending on the value of alternative). The observed test statistics are obs[1,]*obs[3,]/obs[2,].
`rawp`	Numeric vector of unadjusted ("raw") marginal p-values.
`alternative`	Character string indicating the alternative hypotheses, by default 'two.sided'. For one-sided tests, use 'less' or 'greater' for null hypotheses of 'greater than or equal' (i.e. alternative is 'less') and 'less than or equal', respectively.
`get.cutoff`	Logical indicating whether to compute thresholds for the test statistics. Default is 'FALSE'.
`get.cr`	Logical indicating whether to compute confidence intervals for the estimates. Not available for f-tests. Default is 'FALSE'.
`get.adjp`	Logical indicating whether to compute adjusted p-values. Default is 'TRUE'.
`alpha`	The target nominal type I error rate, which may be a vector of error rates. Default is 0.05.

Details

Having selected a suitable test statistics null distribution, there remains the main task of specifying rejection regions for each null hypothesis, i.e., cut-offs for each test statistic. One usually distinguishes between two main classes of multiple testing procedures, single-step and stepwise procedures. In single-step procedures, each null hypothesis is evaluated using a rejection region that is independent of the results of the tests of other hypotheses. Improvement in power, while preserving Type I error rate control, may be achieved by stepwise (step-down or step-up) procedures, in which rejection of a particular null hypothesis depends on the outcome of the tests of other hypotheses. That is, the (single-step) test procedure is applied to a sequence of successively smaller nested random (i.e., data-dependent) subsets of null hypotheses, defined by the ordering of the test statistics (common cut-offs or maxT procedures) or unadjusted p-values (common-quantiles or minP procedures).

In step-down procedures, the hypotheses corresponding to the most significant test statistics (i.e., largest absolute test statistics or smallest unadjusted p-values) are considered successively, with further tests depending on the outcome of earlier ones. As soon as one fails to reject a null hypothesis, no further hypotheses are rejected. In contrast, for step-up procedures, the hypotheses corresponding to the least significant test statistics are considered successively, again with further tests depending on the outcome of earlier ones. As soon as one hypothesis is rejected, all remaining more significant hypotheses are rejected.

These functions perform the following procedures:
ss.maxT: single-step, common cut-off (maxima of test statistics)
ss.minP: single-step, common quantile (minima of p-values)
sd.maxT: step-down, common cut-off (maxima of test statistics)
sd.minP: step-down, common quantile (minima of p-values)

Value

A list with the following components:

`c`	Object of class `"matrix"`, for each nominal (i.e. target) level for the test, a vector of threshold values for the vector of test statistics.
`cr`	Object of class `"array"`, for each nominal (i.e. target) level for the test, a matrix of lower and upper confidence bounds for the parameter of interest for each hypothesis. Not available for f-tests.
`adjp`	Object of class `"numeric"`, adjusted p-values for each hypothesis.

Author(s)

Katherine S. Pollard with design contributions from Sandrine Dudoit and Mark J. van der Laan.

References

M.J. van der Laan, S. Dudoit, K.S. Pollard (2004), Augmentation Procedures for Control of the Generalized Family-Wise Error Rate and Tail Probabilities for the Proportion of False Positives, Statistical Applications in Genetics and Molecular Biology, 3(1). `http://www.bepress.com/sagmb/vol3/iss1/art15/`

M.J. van der Laan, S. Dudoit, K.S. Pollard (2004), Multiple Testing. Part II. Step-Down Procedures for Control of the Family-Wise Error Rate, Statistical Applications in Genetics and Molecular Biology, 3(1). `http://www.bepress.com/sagmb/vol3/iss1/art14/`

S. Dudoit, M.J. van der Laan, K.S. Pollard (2004), Multiple Testing. Part I. Single-Step Procedures for Control of General Type I Error Rates, Statistical Applications in Genetics and Molecular Biology, 3(1). `http://www.bepress.com/sagmb/vol3/iss1/art13/`

Katherine S. Pollard and Mark J. van der Laan, "Resampling-based Multiple Testing: Asymptotic Control of Type I Error and Applications to Gene Expression Data" (June 24, 2003). U.C. Berkeley Division of Biostatistics Working Paper Series. Working Paper 121. `http://www.bepress.com/ucbbiostat/paper121`

See Also

`MTP`

Examples

```
## These functions are used internally by the MTP function
## See MTP function: ? MTP
```

---

wapply — *Weighted version of the apply function*

Description

A function to perform 'apply' on an matrix of data and corresponding matrix of weights.

Usage

```
wapply(X, MARGIN, FUN, W = NULL, ...)
```

Arguments

X	A matrix of data.
MARGIN	A vector giving the subscripts which the function will be applied over. 1 indicates rows, 2 indicates columns.
FUN	The function to be applied. In the case of functions like + the function name must be quoted.
W	An optional matrix of weights. When W=NULL, the usual apply function is called.
...	optional arguments to FUN.

Details

When weights are provided, these are passed to FUN along with the data X. For example, if FUN=meanX, each data value is multiplied by the corresponding weight before the mean is applied.

Value

If each call to FUN returns a vector of length n, then wapply returns an array of dimension c(n, dim(X)[MARGIN]) if n > 1. If n = 1, wapply returns a vector if MARGIN has length 1 and an array of dimension dim(X)[MARGIN] otherwise. If n = 0, the result has length 0 but not necessarily the "correct" dimension.

If the calls to FUN return vectors of different lengths, wapply returns a list of length dim(X)[MARGIN].

This function is used in the package multtest to compute weighted versions of test statistics. It is called by the function get.Tn inside the user-level function MTP.

Author(s)

Katherine S. Pollard

See Also

get.Tn, MTP

Examples

```
data<-matrix(rnorm(200),nr=20)
weights<-matrix(rexp(200,rate=0.1),nr=20)
wapply(X=data,MARGIN=1,FUN=mean,W=weights)
```

Index