multtest

April 19, 2009

`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
nulldist = ..., 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`.

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"`), 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).

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

seed: An integer 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"`). 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 - alternative, typeone, k, q, fdr.method, alpha, smooth.null, method, get.cr, get.cutoff, get.adjp, keep.nulldist. 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 a non-empty nulldist slot (ie: must have been called with 'keep.nulldist=TRUE').

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-methods, [-methods, as.list-methods, print-methods, plot-methods, summary-methods

Examples

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

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

Description

Summary, printing, plotting, subsetting, updating and as.list methods were defined for the MTP class. 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 an object of class MTP.

Methods

[: 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 - alternative, typeone, k, q, fdr.method, alpha, smooth.null, method, get.cr, get.cutoff, get.adjp, keep.nulldist. 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 a non-empty nulldist slot (ie: must have been called with 'keep.nulldist=TRUE').

Author(s)

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

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 each test, 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. 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", csnull=TRUE,B = 1000, method = "ss.maxT", get.cr = FAL
get.cutoff = FALSE, get.adjp = TRUE, keep.nulldist = TRUE, seed = NULL, cluster=
type = NULL, dispatch = 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 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 appropraiately. 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 conrtolled.

`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 kernal 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 non-parametric bootstrap ('boot').

`csnull` Indicator of whether the bootstrap estimated test statistics distribution should be centered and scaled (to produce a null distirbution) or not. If `csnull==FALSE`, the non-null bootstrap estimated test statistics distribution is returned.

`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.

`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 null distribution, by default 'TRUE'. Note that this matrix can be quite large.

`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.

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. 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`.

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 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).

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.

Value

An object of class `MTP`, with the following slots:

`this-is-escaped-codenormal-bracket171bracket-normal`
: Object of class `numeric`, observed test statistics for each hypothesis, specified by the values of the `MTP` arguments `test`, `robust`, `standardize`, and `psi0`.

`this-is-escaped-codenormal-bracket180bracket-normal`
: 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.

`this-is-escaped-codenormal-bracket183bracket-normal`
: Object of class `numeric`, number of columns (i.e. observations) in the input data set.

this-is-escaped-codenormal-bracket187bracket-normal
: Object of class numeric, unadjusted, marginal p-values for each hypothesis.

this-is-escaped-codenormal-bracket191bracket-normal
: Object of class numeric, adjusted (for multiple testing) p-values for each hypothesis (computed only if the get.adjp argument is TRUE).

this-is-escaped-codenormal-bracket196bracket-normal
: 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).

this-is-escaped-codenormal-bracket201bracket-normal
: 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).

this-is-escaped-codenormal-bracket206bracket-normal
: Object of class "matrix", rejection indicators (TRUE for a rejected null hypothesis), for each value of the nominal Type I error rate alpha.

this-is-escaped-codenormal-bracket211bracket-normal
: 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"), 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).

this-is-escaped-codenormal-bracket218bracket-normal
: Object of class call, the call to the MTP function.

this-is-escaped-codenormal-bracket222bracket-normal
: 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"). See ? set.seed for details.

Note

Thank you to Peter Dimitrov for suggestions about the code.

Author(s)

Katherine S. Pollard 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/

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-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 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)
```

---

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.

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

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[

mt.plot(allp, teststat, plottype="rvsa", proc=c("rawp","Bonferroni","maxT","minP"),leg=c(
mt.plot(allp, teststat, plottype="pvsr", proc=c("rawp","Bonferroni","maxT","minP"),leg=c(
mt.plot(allp, teststat, plottype="pvst", proc=c("rawp","Bonferroni","maxT","minP"),leg=c(

# Permutation adjusted p-values for minP procedure with F-statistics (like equal variance
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="pairt")

# 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
```

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=

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

`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).

Usage

```
mt.rawp2adjp(rawp, proc=c("Bonferroni", "Holm", "Hochberg", "SidakSS", "SidakSD"
```

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"`.

Details

Adjusted p-values are computed for simple FWER and FDR controlling procedures based on a vector of raw (unadjusted) p-values.

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).

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),]`.

Author(s)

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

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 D. Yekutieli (2001). The control of the false discovery rate in multiple hypothesis testing under dependency. *Annals of Statistics*. Accepted.

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

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")
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.
mt.sample.rawp(V,classlabel,test="t",side="abs",fixed.seed.sampling="y",B=10000,
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="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.

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")
```

Index