globaltest

October 25, 2011

`Comparative proportions`

Comparative proportions for the Global Test

Description

Comparing the result of a global test performed on a subset to the test results of random subsets of the same size.

Usage

```
comparative(object, N = 1e3, z.scores = TRUE, trace)
```

Arguments

`object`	A `gt.object`, usually one in which one or more subsets of a large number of covariates were tested.
`N`	The number of random subsets to generate.
`z.scores`	If set to `TRUE`, compares the subset to random subsets on the basis of the z-scores of the test. If `FALSE`, uses the p-values instead.
`trace`	If set to `TRUE`, reports progress information. The default is to set trace to `TRUE` if R is in `interactive` mode and more than one comparative proportion is to be calculated.

Details

In a situation when many covariates out of a large set are associated with the response, it is sometimes interesting to know p-value of the tested subset compares to random subsets of the same size. The `comparative` function calculates the proportion of random subsets of the covariates of the same size as the tested subset that have a better score than the tested subset. This proportion is a diagnostic tool to help interpret the test result; it should not be interpreted as a p-value.

Value

An object of class `gt.object` with an appropriate column added to the test results matrix.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

References

The comparative proportion is an enrichment type analysis. For the pros and cons of such an analysis, see

Goeman and Buhlmann (2007) Analyzing gene expression data in terms of gene sets: methodological issues. Bioinformatics 23 (8) 980-987.

See Also

The `gt` function. The `gt.object` function and useful functions associated with that object.

Examples

```
# Simple examples with random data here
# Real data examples in the Vignette

# Random data: covariates A,B,C are correlated with Y
set.seed(1)
Y <- rnorm(20)
X <- matrix(rnorm(200), 20, 10)
X[,1:3] <- X[,1:3] + Y
colnames(X) <- LETTERS[1:10]

# Some subsets of interest
my.sets <- list(c("A", "B"), c("C","D"), c("D", "E"))

# Comparative proportions
res <- gt(Y, X, subsets = my.sets)
comparative(res)
comparative(res, z.scores=FALSE)
```

`Diagnostic plots for globaltest`
Global Test diagnostic plots

Description

Plots to visualize the result of a Global Test in terms of the contributions of the covariates and the subjects.

Usage

```
covariates(object,
              what = c("p-value", "statistic", "z-score", "weighted"),
              cluster = "average", alpha = 0.05, sort = TRUE, zoom = FALSE,
              legend = TRUE, colors, alias, help.lines = FALSE,
              cex.labels = 0.6, pdf, trace)

features(...)

subjects(object,
              what = c("p-value", "statistic", "z-score", "weighted"),
              cluster = "average", sort = TRUE, mirror = TRUE,
              legend = TRUE, colors, alias, help.lines = FALSE,
              cex.labels = 0.6, pdf)
```

Arguments

`object`	A `gt.object`, usually created by a call to `gt`. The object must contain only a single test result, unless the `pdf` argument is used. See the help page of `gt.object` on reducing such an object in case it contains more than one test.
`what`	Gives a choice between various presentations of the same plot. See below under details.
`cluster`	The type of hierarchical clustering performed for the dendrogram. Default is average linkage clustering. For other options, see `hclust`. Setting `cluster = "none"` or `cluster = FALSE` suppresses the dendrogram altogether.
`alpha`	Parameter between 0 and 1. Sets the level of family-wise error control in the multiple testing procedure performed on the dendrogram. See below under details.
`sort`	If `TRUE`, the plot sorts the bars with the most significant covariates and subjects to the left, as far as is possible within the constraints of the dendrogram (if present).
`zoom`	If `TRUE`, discards non-significant branches from the dendrogram with the corresponding covariates. This is especially useful for large sets to "zoom" in on the significant results. If no dendrogram is requested, `zoom = TRUE` discards all covariates that are not significant after Holm multiple testing correction.
`legend`	If `TRUE`, draws a legend in the plot. To override the default labels of the legend, `legend` may also be given as a character vector with the labels of the legend.
`colors`	The colors to be used for the bars. See `rgb` for details on color specification.
`alias`	Optional alternative labels for the bars in the plots. Should be a character vector of the same length as the number of covariates or subjects, respectively.
`help.lines`	If `TRUE`, prints grey dotted lines that help connect the dendrogram to the bars.
`cex.labels`	Magnification factor for the x-axis labels.
`pdf`	Optional filename (`character`) of the pdf file to which the plots are to be written. If a filename is provided in `pdf`, many `covariates` or `subjects` plots of multiple tests can be made with a single call to `covariates` or `subjects`, writing the results to a pdf file.
`trace`	If `TRUE`, prints progress information. Note that printing progress information involves printing of backspace characters, which is not compatible with use of Sweave. Defaults to `gt.options()$trace`.
`mirror`	If `TRUE`, plots the reverse of the scores for the subjects with negative residual response, so that "good" scores are positive for all subjects.
`...`	All arguments of `features` are identical to those of `covariates`.

Details

These two diagnostic plots decompose the test statistics into the contributions of the different covariates and subjects to make the influence of these covariates and subjects visible.

The `covariates` plot exploits the fact that the global test statistic for a set of alternative covariates can be written as a weighted sum of the global test statistics for each single contributing covariate. By displaying these component global test results in a bar plot the `covariates` plot gives insight into the subset of covariates that is most responsible for the significant test result. The plot can show the `p-values` of the component tests on a reversed log scale (the default); their test `statistics`, with stripes showing their mean and standard deviation under the null hypothesis; the `z-scores` of these test statistics, standardized to mean zero and standard deviation one; or the `weighted` test statistics, where the test statistics are multiplied by the relative weight that each covariate carries in the overall test. See the Vignette for more details.

The dendrogram of the `covariates` plot is based on correlation distance if the `directional` argument was set to `TRUE` in the call to `gt`, and uses absolute correlation distance otherwise. The coloring of the dendrogram is based on the multiple testing procedure of Meinshausen (2008): this procedure controls the family-wise error rate on all `2n-1` hypotheses associated with the subsets of covariates induced by the clustering graph. All significant subsets are colored black; non-significant ones remain grey. This coloring serves as an additional aid to find the subsets of the covariates most contributing to a significant test result.

The `features` function is a synonym for `covariates`, using exactly the same arguments.

The `subjects` plot exploits the fact that the global test can be written as a sum of contributions of each individual. Each of these contributions is itself a test statistic for the same null hypothesis as the full global test, but one which puts a greater weight on the observed information of a specific subject. These test statistic of subject `i` is significant if, for the other subjects, similarity in the alternative covariates to subject `i` tends to coincide with similarity in residual response to subject `i`. Like the `covariates` plot, the `subjects` plot can show the `p-values` of these component tests on a reversed log scale (the default); their test `statistics`, with stripes showing their mean and standard deviation under the null hypothesis; the `z-scores` of these test statistics, standardized to mean zero and standard deviation one; or the `weighted` test statistics, where the test statistics are multiplied by the relative weight that each covariate carries in the overall test. Setting `mirror=FALSE` reverses the bars of subjects with a negative residual response (not applicable if `p-values` are plotted). The resulting `statistics` values have the additional interpretation that they are proportional to the first order estimates of the linear predictors of each subject under the alternative, i.e. subjects with positive values have higher means under the alternative than under the null, and subjects with negative values have lower means under the alternative than under the null. See the Vignette for more details.

The dendrogram of the `subjects` plot is always based on correlation distance. There is no analogue to Meinshausen's multiple testing method for this dendrogram, so multiple testing is not performed.

Value

If called to make a single plot, the `covariates` function returns an object of class `gt.object`. Several methods are available to access this object: see `gt.object`. The `subjects` function returns a matrix. If called to make multiple plots, both functions return `NULL`.

Note

The term "z-score" is not meant to imply a normal distribution, but just refers to a studentized score. The z-scores of the `subjects` plot are asymptotically normal under the null hypothesis; the z-scores of the `covariates` plot are asymptotically distributed as a chi-squared variable with one degree of freedom.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Livio Finos.

References

General theory and properties of the global test are described in

Goeman, Van de Geer and Van Houwelingen (2006) Journal of the Royal Statistical Society, Series B 68 (3) 477-493.

Meinshausen's method for multiple testing

Meinshausen (2008) Biometrika 95 (2) 265-278.

For more references related to applications of the test, see the vignette GlobalTest.pdf included with this package.

See Also

Diagnostic plots: `covariates`, `subjects`.

The `gt.object` function and useful functions associated with that object.

Many more examples in the vignette!

Examples

```
# Simple examples with random data here
# Real data examples in the Vignette

# Random data: covariates A,B,C are correlated with Y
set.seed(1)
Y <- rnorm(20)
X <- matrix(rnorm(200), 20, 10)
X[,1:3] <- X[,1:3] + Y
colnames(X) <- LETTERS[1:10]

# Preparation: test
res <- gt(Y,X)

# Covariates
covariates(res)
covariates(res, what = "w")
covariates(res, zoom = TRUE)

# Subjects
subjects(res)
subjects(res, what = "w", mirror = FALSE)

# Change legend, colors or labels
covariates(res, legend = c("upregulated", "downregulated"))
covariates(res, col = rainbow(2))
covariates(res, alias = letters[1:10])
```

`gt gene set testing methods`
Gene set testing of gene set databases using Global Test

Description

A collection of procedures for performing the Global Test on gene set databases. Three function are provided for KEGG, for Gene Ontology and for the Broad Institute's gene sets.

Usage

```
gtKEGG (response, exprs, ..., id, annotation, probe2entrez,
           multtest = c("Holm", "BH", "BY"), sort = TRUE)

gtGO (response, exprs, ..., id, annotation, probe2entrez,
           ontology = c("BP", "CC", "MF"), minsize=1, maxsize=Inf,
           multtest = c("Holm", "focuslevel", "BH", "BY"),
           focuslevel = 10, sort = TRUE)


gtConcept (response, exprs, ..., annotation, probe2entrez,
           conceptmatrix, concept2name = "conceptID2name.txt",
           entrez2concept = "entrezGeneToConceptID.txt", threshold = 1e-4,
           share = TRUE, multtest = c("Holm", "BH", "BY"),
           sort = TRUE)

gtBroad (response, exprs, ..., id, annotation, probe2entrez, collection,
           category = c("c1", "c2", "c3", "c4", "c5"),
           multtest = c("Holm", "BH", "BY"), sort = TRUE)
```

Arguments

`response`	The response variable of the regression model. This is passed on to the `response` argument of `gt`.
`exprs`	The expression measurements. May be `ExpressionSet` or matrix. Passed on to the `alternative` argument of `gt`.
`...`	Any other arguments are also passed on to `gt`.
`id`	The identifier(s) of gene sets to be tested (character vector). If omitted, tests all gene sets in the database.
`annotation`	The name of the probe annotation package for the microarray that was used, or the name of the genome wide annotation package for the species (e.g. org.Hs.eg.db for human). If an organism package is given, the argument `probe2entrez` must be supplied. If `annotation` is missing, the function will attempt to retrieve the annotation information from the `exprs` argument.
`probe2entrez`	Use only if no probe annotation package is available. A mapping from probe identifiers to entrez gene ids. May be an environment, named list or named vector.
`multtest`	The method of multiple testing correction. Choose from: Benjamini and Hochberg FDR control (BH); Benjamini and Yekutieli FDR control (BY) or Holm familywise error fontrol (Holm). For `gtGO` also the focus level method is available. See `focusLevel`.
`sort`	If `TRUE`, sorts the results to increasing p-values.
`ontology`	The ontology or ontologies to be used. Default is to use all three ontologies.
`minsize`	The minimum number of probes that may be annotated to a gene set. Gene sets with fewer annotated probes are discarded.
`maxsize`	The maximum number of probes that may be annotated to a gene set. Gene sets with more annotated probes are discarded.

`focuslevel`	The focus level to be used for the focus level method. Either a vector of gene set ids, or a numerical level. In the latter case, `findFocus` is called with `maxsize` at the specified level to find a focus level.
`collection`	The Broad gene set collection, created by a call to `getBroadSets`.
`conceptmatrix`	The name of the file containing the importance weights, i.e. concept profile associations between Anni concepts. In the matrix contained in the file, columns correspond to testable concepts, and rows correspond to entrez-concepts. Useable files can be downloaded from `http://www.biosemantics.org/weightedglobaltest`.
`concept2name`	The name of the file containing a mapping between Anni concepts and entrez identifiers. Useable files can be downloaded from `http://www.biosemantics.org/weightedglobaltest`.
`entrez2concept`	The name of the file containing a mapping between Anni concept numbers and names. Useable files can be downloaded from `http://www.biosemantics.org/weightedglobaltest`.
`threshold`	The relevance threshold for importance weights. Importance weights below the threshold are treated as zero.
`share`	If `TRUE`, the function divides the importance weight of a gene over all probes corresponding to the same entrez identifier. If `FALSE`, all probes get the full importance weight of the gene.
`category`	The subcategory of the Broad collection to be tested. The default is to test all sets.

Details

These are utility functions to make it easier to do gene set testing of gene sets available in gene set databases. The functions automatically retrieve the gene sets, preprocess and select them, perform global test, do multiple testing correction, and sort the results on the basis of their p-values.

The four functions use different databases for testing. `gtKEGG` and `gtGO` use KEGG (`http://www.genome.jp/kegg`) and GO (`http://www.geneontology.org`); `gtConcept` uses the Anni database (`http://www.biosemantics.org/anni`), and `gtBroad` uses the MSigDB database (`http://http://www.broadinstitute.org/gsea/msigdb`). The `gtConcept` function differs from the other three in that it uses association weights between 0 and 1 for genes within sets, rather than having a hard cut-off for membership of a gene in a set.

All functions require that `annotate` and the appropriate annotation packages are installed. `gtKEGG` additionally requires the `KEGG.db` package; `gtGO` requires the `GO.db` package; `gtBroad` requires the user to download the XML file "msigdb_v2.5.xml" from \`http://www.broad.mit.edu/gsea/downl` and to preprocess that file using the `getBroadSets` function. `gtConcept` requires files that can be downloaded from `http://www.biosemantics.org/weightedglobaltest`.

Value

A `gt.object`.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

References

Goeman, Van de Geer, De Kort and Van Houwelingen (2004). A global test for groups of genes: testing association with a clinical outcome. Bioinformatics 20 (1) 93-99.

Goeman, Oosting, Cleton-Jansen, Anninga and Van Houwelingen (2005). Testing association of a pathway with survival using gene expression data. Bioinformatics 21 (9) 1950-1957.

See Also

The `gt` function. The `gt.object` and useful functions associated with that object.

Examples

```
# Examples in the Vignette
```

`gt goodness of fit methods` *Goodness of fit testing in regression models using Global Test*

Description

Tests the goodness of fit of a regression model against a specified alternative using the Global Test. Three main functions are provided: `gtPS` uses Penalized Splines, `gtK` uses Kernel smoothers and `gtI` uses linear Interactions. The other functions are for external use in combination with `gt`.

Usage

```
gtPS(response, null, data, model, ..., covs, bdeg = 3, nint= 10, pord = 2,
interact = FALSE, by = NULL, robust = FALSE, termlabels = FALSE)

gtK(response, null, data, model, ..., covs, quant = .25,
       metric = c("euclidean", "pearson"),
       kernel=c("uniform", "exponential", "triangular", "neighbours", "Gauss"),
       robust = FALSE, scale = TRUE, termlabels = FALSE)

gtI(response, null, data, model, ..., covs, iorder=2)

bbase(x, bdeg, nint)

repdes(B, pord, K=NULL, tol = 1e-10)

getPS(data, covs, bdeg=3, nint=10, pord=2, by=NULL, interact=FALSE)

getK(data, covs, quant=.25, metric = c("euclidean", "pearson"),
kernel=c("uniform", "exponential", "triangular","neighbours","Gauss"),
scale=TRUE )
```

Arguments

`response`	The response vector of the regression model. May be supplied as a vector, as a `formula` object, or as an object of class `lm`, `glm` or `coxph`. In the last two cases, the specification of `null` is not required.
`null`	The null design matrix. May be given as a matrix or as a half `formula` object (e.g. `~a+b`).
`data`	Only used when `response` or `null` is given in formula form. An optional data frame, list or environment containing the variables used in the formulae.
`model`	The type of regression model to be tested. If omitted, the function will try to determine the model from the class and values of the `response` argument.
`...`	Any other arguments are also passed on to `gt`.
`covs`	A variable or a vector of variables that are the covariates the smooth terms are function of.
`bdeg`	A vector or a list of vectors which specifies the degree of the B-spline basis.
`nint`	A vector or a list of vectors which specifies the number of intervals determined by equally-spaced knots.
`pord`	A vector or a list of vectors which specifies the order of the differences indicating the type of the penalty imposed to the coefficients.
`interact`	`TRUE` to consider a multidimensional smooth function of `covs`.
`by`	A factor variable or a vector of factor variables to let the smooth terms specified in `covs` interact with factors terms.
`termlabels`	`TRUE` to consider e.g. `s(log(cov))` instead of `s(cov)` when `null=~ log(cov)` and `covs` is missing.
`robust`	`TRUE` to obtain an overall test which combines multiple specifications of the B-spline basis arguments (when `bdeg`, `nint` and `pord` are lists) or multiple specifications of the bandwidth (when `quant` is a vector of quantiles).
`quant`	The smoothing bandwidth to be used, expressed as the percentile of the distribution of distance between observations, with default the 25th percentile. To investigate the sensitivity to different choices, `quant` can be a vector of percentiles. See also `robust` argument.
`metric`	A character string specifying the metric to be used. The available options are "euclidean" (the default), "pearson" and "mixed" (to be implemented). "mixed" distance is chosen automatically if some of the selected covariates are not numeric.
`kernel`	A character string giving the smoothing kernel to be used. This must be one of "uniform", "exponential", "triangular", "neighbours", or "Gauss", with default "uniform".
`scale`	`TRUE` to scale the covariates before computing the distance.
`iorder`	Order of the linear interactions, e.g. second order interactions, third order etc.
`x`	A numeric vector of values at which to evaluate the B-spline basis
`B`	Matrix containing the B-spline basis
`K`	Penalty matrix (i.e. the penalty term is the quadratic form of K and the spline coefficients)
`tol`	Eigenvalues smaller than `tol` are considered zero

Details

These are functions to test for specific types of lack of fit by using the Global Test. Suppose that we are concerned with the adequacy of some regression model `response ~ null`, such as `Y ~ X1 + X2`. The alternative model can be cast into the generic form `response ~ null + alternative`, which comprises different models corresponding to different types of lack of fit. Thus, the specification of `alternative` is required. It identifies the type of lack of fit the test is directed against.

By using `gtPS`, the alternative is given by a user specified sum of smooth functions of continuous covariates, e.g. `alternative= ~ s(X1)` when `covs="X1"` and `alternative= ~ s(X1) + s(X2)` when `covs=c("X1","X2")`. Smooth terms are represented using P-splines as proposed by Eilers and Marx (1996). This approach consists in constructing a B-spline basis of degree `bdeg` with `nint + 1` equidistant knots, where a difference penalty of order `pord` is applied to the basis coefficients. Any sane combination of penalty order and basis degree is allowed. If `interact=TRUE`, the alternative is given by a multidimensional smooth function of `covs`, which is represented by a tensor product of the marginal B-splines bases and the Kronecker sum of the marginal penalties, e.g. `alternative= ~ s(X1,X2)` when `covs=c("X1","X2")` and `interact=TRUE`. If the `null` model includes continuous and factor variables, e.g. `X1` continuous and `X2` factor, the argument `by` allows to construct varying-coefficient alternative models, e.g. `alternative= ~ s(X1):X2`, when `covs="X1"` and `by="X2"`. If the response is survival and the `null` model includes factor variables, e.g. `response=Surv(time,status)` and `X2` is a factor, the argument `by` allows to construct time-varying effects alternative models, e.g. `alternative= ~ s(time):X2` by using `covs="time"` and `by="X2"`.

By using `gtK` the alternative is given by a user specified multidimensional smooth term, e.g. `alternative= ~ s(X1, X2)` when `covs=c("X1","X2")`. Multidimensional smooth terms are represented by a kernel smoother defined by a distance measure (`metric`), a kernel shape (`kernel`) and a bandwidth (`quant`). Because the test is sensitive to the chosen value of `quant`, it is possible to specify `quant` as a vector of different values in combination with `robust=TRUE`. Distance measures for factor covariates and for the situation that both continuous and factor covariates are present are constructed as in le Cessie and van Houwelingen (1995), e.g. `covs=c("X1","X2")` and `distance="mixed"` when `X1` continuous and `X2` factor.

By using `gtI`, the alternative is given by all the possible ith-order linear interactions between `covs`, e.g. `alternative= ~ X1:X2` when `covs=c("X1","X2")` and `iorder=2`.

The remaining functions are meant for constructing the alternative design matrix that will be used in the `alternative` argument of the `interact=TRUE` function. `bbase` constructs the B-spline basis for the covariate `cov` and `ndiff` builts a difference penalty matrix of order `pord`. These function are based on the functions provided by Eilers and Marx (1996). `repdes` reparameterizes a spline basis `B` via the spectral decomposition of its penalty matrix `K` into a design for the unpenalized part of the function and a design for the penalized part with i.i.d. coefficients. `getPS` and `getK` return a reparameterized B-spline basis and a Kernel smoothing matrix, respectively.

See the vignette for more examples.

Value

The function returns an object of class `gt.object`. Several operations and diagnostic plots can be made from this object.

Note

Currently linear (normal), logistic, multinomial logistic and Poisson regression models with canonical links and Cox's proportional hazards regression model are supported.

Author(s)

Aldo Solari: `<aldo.solari@unimib.it>`

References

Eilers, Marx (1996). Flexible smoothing with B-splines and penalties. Statistical Science, 11: 89-121.

le Cessie, van Houwelingen (1995). Testing the Fit of a Regression Model Via Score Tests in Random Effects Models. Biometrics 51: 600-614.

For references related to applications of the test, see the vignette GlobalTest.pdf included with this package.

See Also

The `gt` function. The `gt.object` and useful functions associated with that object.

Examples

```
# Random data: univariate
set.seed(0)
X1<-runif(50)
s1 <- function(x) exp(2 * x)
e <- rnorm(50)
Y <-  s1(X1) + e

### gtPS
gtPS(Y~X1)

# model input
rdata<-data.frame(Y,X1)
nullmod<-lm(Y~X1,data=rdata)
gtPS(nullmod)

# formula input and termlabels
gtPS(Y~exp(X1),data=rdata)
gtPS(Y~exp(X1),covs="exp(X1)",data=rdata)
gtPS(Y~exp(X1),data=rdata, termlabels=TRUE)

# P-splines arguments
gtPS(Y~X1, nint=list(a=2, b=10, c=50))
gtPS(Y~X1, nint=list(a=2, b=10, c=50), pord=0)
gtPS(Y~X1, nint=list(a=10, b=10), pord=list(a=0,b=2))
gtPS(Y~X1, nint=list(a=10, b=10), pord=list(a=0,b=2), robust=TRUE)

# Random data: additive model
X2<-runif(50)
s2 <- function(x) 0.2 * x^11 * (10 * (1 - x))^6 + 10 * (10 * x)^3 * (1 - x)^10
Y <-  s1(X1) + s2(X2) + e
gtPS(Y~X1+X2)
gtPS(Y~X1+X2, pord=list(a=c(0,2), b=c(2,0)))
gtPS(Y~X1+X2, covs="X2")

### gtK
gtK(Y~X1+X2)
gtK(Y~X1+X2, quant=seq(.05,.95,.1))
gtK(Y~X1+X2, quant=seq(.05,.95,.1), robust=TRUE)

# Random data: smooth surface
s12 <- function(a, b, sa = 1, sb = 1) {
        (pi^sa * sb) * (1.2 * exp(-(a - 0.2)^2/sa^2 - (b - 0.3)^2/sb^2) +
        0.8 * exp(-(a - 0.7)^2/sa^2 - (b - 0.8)^2/sb^2))
        }
Y <- s12(X1,X2) + e

# Non-linear interaction
gtPS(Y~X1*X2)
gtPS(Y~X1*X2, interact=TRUE)
gtK(Y~X1*X2, quant=seq(.05,.95,.1), robust=TRUE)

# Time-varying effects
require("survival")
data(rats)
gtPS(Surv(time,status)~rx,covs="time", by="rx",data=rats, model="cox", pord=1)
```

---

gt — *Global Test (new implementation, 2009)*

Description

Tests a low-dimensional null hypothesis against a potentially high-dimensional alternative in regression models (linear regression, logistic regression, poisson regression, Cox proportional hazards model).

Usage

```
gt(response, alternative, null, data, test.value,
     model = c("linear", "logistic", "cox", "poisson", "multinomial"), levels,
     directional = FALSE, standardize = FALSE, permutations = 0, subsets,
     weights, alias, x = FALSE, trace)
```

Arguments

`response`	The response vector of the regression model. May be supplied as a vector or as a `formula` object. In the latter case, the right hand side of `response` is passed on to `alternative` if that argument is missing, or otherwise to `null`.
`alternative`	The part of the design matrix corresponding to the alternative hypothesis. The covariates of the null model do not have to be supplied again here. May be given as a half `formula` object (e.g. `~a+b`). In that case the intercept is always suppressed. Alternatively, the `alternative` argument may also be given as an `ExpressionSet` object, in which case `t(exprs(alternative))` is used for the `alternative` argument, and `pData(alternative)` is passed on to the `data` argument if that argument is missing.
`null`	The part of the design matrix corresponding to the null hypothesis. May be given as a design matrix or as a half `formula` object (e.g. `~a+b`). The default for `null` is `~1`, i.e. only an intercept. This intercept may be suppressed, if desired, with `null = ~0`.

`data`	Only used when `response`, `alternative`, or `null` is given in formula form. An optional data frame, list or environment containing the variables used in the formulae. If the variables in a formula are not found in `data`, the variables are taken from environment(formula), typically the environment from which `gt` is called.
`test.value`	An optional vector regression coefficients to test. The default is to test the null hypothesis that all regression coefficients of the covariates of the alternative are zero. The `test.value` argument can be used to test a value other than zero. The coefficients are applied to the design matrix of `alternative` before any standardization (see the `standardize` argument).
`model`	The type of regression model to be tested. If omitted, the function will try to determine the model from the class and values of the `response` argument.
`levels`	Only used if response is `factor`. Selects a subset of `levels(response)` to be tested, given as a character vector. If a vector of length `>1`, the test uses only the subjects with the specified outcome categories. If `levels` is of length 1, the test reduces the response to a two-valued factor, testing the specified outcome category against the others combined.
`directional`	If set to `TRUE`, directs the power of the test especially against the alternative that the true regression coefficients under the alternative have the same sign. The default is that the power of the test does not depend on the sign of the true regression coefficients. Set negative `weights` for covariates that are expected to have opposite sign.
`standardize`	If set to `TRUE`, standardizes all covariates of the alternative to have unit second central moment. This makes sure that the test result is independent of the relative scaling of the covariates. The default is to let covariates with more variance have a greater weight in the test.
`permutations`	The number of permutations to use. The default, `permutations = 0`, uses the asymptotic distribution. The asymptotic distribution is the exact distribution in case of the linear model with normal errors.
`subsets`	Optional argument that can be used to test one or more subsets of the covariates in `alternative`. Can be a vector of column names or column indices of `alternative`, or a list of such vectors. In the latter case, a separate test will be performed for each subset.
`weights`	Optional argument that can be used to give certain covariates in `alternative` greater weight in the test. Can be a vector or a list of vectors. In the latter case, a separate test will be performed for each weight vector. If both `subsets` and `weights` are specified as a list, they must have the same length. In that case, `weights` vectors may have either the same length as the number of covariates in `alternative`, or the same length as the corresponding subset vector. Weights can be negative; the sign has no effect unless `directional` is `TRUE`.
`alias`	Optional second label for each test. Should be a vector of the same length as `subsets`. See also `alias`.
`x`	If `TRUE`, gives back the `null` and `alternative` design matrices. Default is not to return these matrices.
`trace`	If `TRUE`, prints progress information. This is useful if many tests are performed, i.e.\ if `subsets` or `weights` is a list. Note that printing progress information involves printing of backspace characters, which is not compatible with use of Sweave. Defaults to `gt.options()$trace`.

Details

The Global Test tests a low-dimensional null hypothesis against a (potentially) high-dimensional alternative, using the locally most powerful test of Goeman et al (2006). In this regression model implementation, it tests the null hypothesis `response ~ null`, that the covariates in `alternative` are not associated with the response, against the alternative model `response ~ null + alternative` that they are.

The test has a wide range of applications. In gene set testing in microarray data analysis `alternative` may be a matrix of gene expression measurements, and the aim is to find which of a collection of predefined `subsets` of the genes (e.g. Gene Ontology terms or KEGG pathways) is most associated with the `response`. In penalized regression or other machine learning techniques, `alternative` may be a collection of predictor variables that may be used to predict a `response`, and the test may function as a useful pre-test to see if training the classifier is worthwhile. In goodness-of-fit testing, `null` may be a model with linear terms fitted to the `response`, and `alternative` may be a large collection of non-linear terms. The test may be used in this case to test the fit of the null model with linear terms against a non-linear alternative.

See the vignette for extensive examples of these applications.

Value

The function returns an object of class `gt.object`. Several operations and diagnostic plots can be made from this object. See also Diagnostic plots.

Note

If `null` is supplied as a `formula` object, an intercept is automatically included. As a consequence `gt(Y, X, Z)` will usually give a different result from `gt(Y, X, ~Z)`. The first call is equivalent to `gt(Y, X, ~0+Z)`, whereas the second call is equivalent to `gt(Y, X, cbind(1,Z))`.

P-values from the asymptotic distribution are accurate to at least two decimal places up to a value of around `1e-12`. Lower p-values are numerically less reliable.

Missing values are allowed in the `alternative` matrix only. Missing values are imputed conservatively (i.e. under the null hypothesis). Covariates with many missing values get reduced variance and therefore automatically carry less weight in the test result.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

References

General theory and properties of the global test are described in

Goeman, Van de Geer and Van Houwelingen (2006) Journal of the Royal Statistical Society, Series B 68 (3) 477-493.

For references related to applications of the test, see the vignette GlobalTest.pdf included with this package.

See Also

Diagnostic plots: `covariates`, `subjects`.

The `gt.object` function and useful functions associated with that object.

Many more examples in the vignette!

Examples

```
# Simple examples with random data here
# Real data examples in the Vignette

# Random data: covariates A,B,C are correlated with Y
set.seed(1)
Y <- rnorm(20)
X <- matrix(rnorm(200), 20, 10)
X[,1:3] <- X[,1:3] + Y
colnames(X) <- LETTERS[1:10]

# Compare the global test with the F-test
gt(Y, X)
anova(lm(Y~X))

# Using formula input
res <- gt(Y, ~A+B, null=~C+E, data=data.frame(X))
summary(res)

# Beware: null models with and without intercept
Z <- rnorm(20)
summary(gt(Y, X, null=~Z))
summary(gt(Y, X, null=Z))

# Logistic regression
gt(Y>0, X)

# Subsets and weights (1)
my.sets <- list(c("A", "B"), c("C","D"), c("D", "E"))
gt(Y, X, subsets = my.sets)
my.weights <- list(1:2, 2:1, 3:2)
gt(Y, X, subsets = my.sets, weights=my.weights)

# Subsets and weights (2)
gt(Y, X, subset = c("A", "B"))
gt(Y, X, subset = c("A", "A", "B"))
gt(Y, X, subset = c("A", "A", "B"), weight = c(.5,.5,1))

# Permutation testing
summary(gt(Y, X, perm=1e4))
```

`gt.object class` *Class "gt.object" for storing the result of the function gt*

Description

The class gt.object is the output of a call to `gt`. It stores the information needed for various diagnostic plots.

Slots

These slots are not meant to be directly accessed by the user.

`result`**:** Object of class "matrix". The number of rows of this matrix is the number of tests performed. The matrix has at least the columns "p-value", "Statistic" "Expected", "Std.dev", and "#Cov".

`extra`**:** Object of class "data.frame". Holds additional information that may be added later about the tests performed, such as multiplicity-adjusted p-values (see `p.adjust`), alias names for tests and comparative proportions (see `comparative`).

`call`**:** The matched call to `gt`.

`functions`**:** A "list" of various functions used by the `covariates` and `subjects` functions and the various methods.

`subsets`**:** A "list" or "NULL". Stores the subsets tested, if more than one.

`structure`**:** A "list" or "NULL". Stores subset and superset relationships between the sets in the "subsets" slot.

`weights`**:** A "list" or "NULL". Stores the weight vectors used for testing, if more than one.

`alternative`**:** If `gt` was called with `x = TRUE`, stores the design matrix of the alternative hypothesis; "NULL" otherwise.

`null`**:** If `gt` was called with `x = TRUE`, stores the design matrix of the null hypothesis; "NULL" otherwise.

`directional` Stores the `directional` argument of the call to `gt`.

`legend` Object of class "list". Stores appropriate legends for the `covariates` and `subjects` plots.

`model` Object of class "character". Stores the model.

Methods

show (gt.object): Prints the test results: p-value, test statistic, expected value of the test statistic under the null hypothesis, standard deviation of the test statistic under the null hypothesis, and number of covariates tested.

summary (gt.object): Prints the test results (as `show`) plus additional information on the model and the test.

p.value (gt.object): Extracts the p-values.

z.score (gt.object): Extracts z-score: (Test statistic - Expected value) / Standard deviation.

result (gt.object): Extracts the results matrix together with the additional (e.g. multiple testing) information in the `extra` slot.

sort (gt.object): Sorts the pathways to increasing p-values. Equal p-values are sorted on decreasing z-scores.

"[" (gt.object): Extracts results of one or more test results if multiple tests were performed. Identical to "[[".

"[[" (gt.object): Extracts results of one or more test results if multiple tests were performed. Identical to "[".

length (gt.object): The number of tests performed.

size (gt.object): Extracts a vector with the number of alternative covariates tested for each test.

names (gt.object): Extracts the row names of the results matrix.

names<- (gt.object): Changes the row names of the results matrix. Duplicate names are not allowed, but see `alias`.

alias (gt.object): Extracts the "alias" column of the results matrix that can be used to add additional information on each test perfomed.

alias<- (gt.object): Changes the "alias" column of the results matrix. Note that unlike for names, duplicate aliases are allowed.

weights (gt.object): extracts the effective weights of the covariates as they are used internally by the test.

subsets (gt.object): extracts the "subsets" slot.

hist (gt.object): Produces a histogram to visualize the permutation test statistics. Only relevant after permutation testing.

covariates (gt.object): Produces a plot to show the influence of individual covariates on the test result. See `covariates` for details.

subjects (gt.object): Produces a plot to show the influence of individual subjects on the test result. See `subjects` for details.

p.adjust (gt.object): Performs multiple testing correction and produces multiplicity-corrected p-values. See `p.adjust` for details.

comparative (gt.object): Compares the p-values of tests performed on a subsets or weights with p-values of random subsets of covariates of same size or randomly distributed weights. See `comparative` for details.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

See Also

`gt`, `covariates`, `subjects`.

Examples

```
# Simple examples with random data here
# Real data examples in the Vignette

# Random data: covariates A,B,C are correlated with Y
Y <- rnorm(20)
X <- matrix(rnorm(200), 20, 10)
X[,1:3] <- X[,1:3] + 0.5*Y
colnames(X) <- LETTERS[1:10]

# Make a gt.object
sets <- list(odd = c(1,3,5,7,9), even = c(2,4,6,8,10))
res <- gt(Y, X, subsets=sets)

# Show the results
res
summary(res)
sort(res)
p.value(res)
subsets(res)

# Names
names(res)
names(res) <- c("ODD", "EVEN")
alias(res) <- c("odd covariates", "even covariates")

# Multiple testing
```

```
p.adjust(res, method = "holm")
p.adjust(res, method = "BH")

# Diagnostics
weights(res)
covariates(res[1])
subjects(res[1])

# Permutation testing
res <- gt(Y, X, perm = 1e4)
hist(res)
```

`gt.options` *Options for globaltest package*

Description

Sets various global options for the functions in the globaltest package.

Usage

```
gt.options (trace, trim, transpose, max.print, warn.deprecated)
```

Arguments

`trace`	(Default: `interactive()`). If `TRUE`, prints progress information whenever many tests are to be performed. Such printing of progress information involves the printing of backspace characters, which is not compatible with use of `Sweave`.
`trim`	(Default: `FALSE`). If `FALSE`, returns an error if covariates in the `subsets` argument of `gt` are not present in the data; if `TRUE`, silently removes these covariates, and remove duplicates.
`transpose`	(Default: `FALSE`). If `TRUE`, `gt` expects the transposed data format that is usual in genomics, in which the subjects correspond to the columns of the data matrix and the covariates (or probes) are the rows. If `FALSE`, `gt` expects the usual statistical format instead.
`max.print`	(Default: `Inf`). The maximum number of characters to print for the alias.
`warn.deprecated`	(Default: `TRUE`). Whether or not to give a warning when the deprecated `globaltest` function is called.

Details

The globaltest options can be set during a session, and apply to all calls to functions in the globaltest package for the rest of the session. They are not remembered between sessions.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`

See Also

The gt function.

Examples

```
# setting options
gt.options(max.print=45, trim=TRUE)

# reading options
gt.options()
```

`mlogit` *Multinomial Logistic Regression*

Description

Fits a multinomial logistic regression model to a nominal scale outcome.

Usage

```
mlogit(formula, data, control = glm.control())
```

Arguments

`formula`	An object of class `formula` containing a symbolic description of the model to be fit. See the documentation of `formula` for details.
`data`	An optional data frame containing the variables in the model. If not found in 'data', the variables are taken from the environment from which 'mlogit' is called.
`control`	A list of parameters for controlling the fitting process. See the documentation of `glm.control` for details.

Details

The function mlogit fits a multinomial logistic regression model for a multi-valued outcome with nominal scale. The implementation and behaviour are designed to mimic those of `glm`, but the options are (as yet) more limited. Missing values are not allowed in the data.

The model is fitted without using a reference outcome category; the parameters are made identifiable by the requirement that the sum of corresponding regression coefficients over the outcome categories is zero.

Value

An object of (S4) class `mlogit`. The class has slots: coefficients (matrix), standard.err (matrix), fitted.values (matrix), x (matrix), y (matrix), formula (formula), call (call), df.null (numeric), df.residual (numeric), null.deviance (numeric), deviance (numeric), iter (numeric), converged (logical).

Methods implemented for the `mlogit` class are `coefficients`, `fitted.values`, `residuals` and which extract the relevant quantities, and `summary`, which gives the same output as with a `glm` object.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

See Also

`glm`, `multinom`.

Examples

```
  y <- factor(rep(1:4, 5))
  x <- 1:20
  fit <- mlogit(y ~ x)
  summary(fit)
  residuals(fit)
```

`Multiple testing methods of the globaltest package`
Multiple testing correction for the Global Test

Description

A collection of multiple testing procedures for the Global Test. Methods for the focus level procedure of Goeman and Mansmann for graph-structured hypotheses, and for the inheritance procedure based on Meinshausen.

Usage

```
# The focus level method:
focusLevel (test, sets, focus, ancestors, offspring,
            stop = 1, atoms = TRUE, trace)

findFocus (sets, ancestors, offspring, maxsize = 10, atoms = TRUE)


# The inheritance method:
inheritance (test, sets, weights, ancestors, offspring, Shaffer,
             homogeneous = TRUE, trace)


# Utilities for focus level and inheritance method:
leafNodes (object, alpha=0.05, type = c("focuslevel","inheritance"))

draw (object, alpha = 0.05, type = c("focuslevel","inheritance"),
      names=FALSE, sign.only = FALSE, interactive = FALSE)
```

Arguments

`object`	A `gt.object`, usually one in which more than one test was performed.
`test`	Either a `function` or `gt.object`. If a function, that function should take as its argument a vector of covariate labels, and return (raw) p-value. See the examples below. If a `gt.object` the call to `gt` that created it must have had all the covariates of `sets` (below) in its `alternative` argument.

`sets`	A named `list` representing covariate sets of the hypotheses of interest, for which adjusted p-values are to be calculated. If it is missing but `test` is a `gt.object`, the `subsets` slot of that object will be used. If used in the `inheritance`, `sets` describe a tree structure of hypotheses. In this case, object of class `hclust` or `dendrogram`.
`focus`	The focus level of the focus level method. Must be a subset of `names(sets)`. Represents the level of the graph at which the method is focused, i.e. has most power.
`ancestors`	An environment or list that maps each set in `sets` to all its ancestors, i.e. its proper supersets. If missing, `ancestors` is determined from the input of `offspring`, or, if that is also missing, from the input of `sets` (time-consuming).
`offspring`	An environment or list that maps each set in `sets` to all its offspring, i.e. its proper subsets. If missing, `offspring` is determined from the input of `ancestors`, or, if that is also missing, from the input of `sets` (time-consuming).
`stop`	Determines when to stop the algorithm. If `stop` is set to a value smaller than or equal to 1, the algorithm only calculates familywise error rate corrected p-values of at most `stop`. If `stop` is set to a value greater than 1, the algorithm stops when it has rejected at least `stop` hypotheses. If set to exactly 1, the algorithm calculates all familywise error rate corrected p-values. Corrected p-values that are not calculated are reported as `NA`.
`atoms`	If set to `TRUE`, the focus level algorithm partitions the offspring of each focus level set into the smallest possible building blocks, called atoms. Doing this often greatly accelerates computation, but sometimes at the cost of some power.
`trace`	If set to `TRUE`, reports progress information. The default is obtained from `gt.options()$trace`. Alternatively, setting `trace = 2` gives much more extensive output (`focusLevel` only).
`maxsize`	Parameter to choose the height of the focus level. The focus level sets are chosen in such a way that the number of tests that is to be done for each focus level set is at most `2^maxsize - 1`.
`alpha`	The alpha level of familywise error control for the significant subgraph.
`Shaffer`	If set to `TRUE`, it applies the Shaffer improvement. If `Shaffer` is `NULL` and `object` is a `gt.object` the procedure checks whether `Shaffer=TRUE` is valid, and sets the value accordingly.
`weights`	Optional weights vector for the leaves nodes. If it is missing but `test` is a `gt.object`, the result of `weights(object)` will be used. In all other cases `weights` is set to be uniform among all leaf nodes.
`homogeneous`	If set to `TRUE`, redistributes the alpha of rejected leaf node hypotheses homogeneously over the hypotheses under test, rather than to closest related hypotheses.
`type`	Argument for specifying which multiple testing correction method should be used. Only relevant if both the inheritance and the focuslevel procedures were performed on the same set of test results.
`names`	If set to `TRUE`, draws the graph with node names rather than numbers.
`sign.only`	If set to `TRUE`, draws only the subgraph corresponding to the significant nodes. If `FALSE`, draws the full graph with the non-significant nodes grayed out.
`interactive`	If set to `TRUE`, creates an interactive graph in which the user can see the node label by clicking on the node.

Details

Multiple testing correction becomes important if the Global Test is performed on many covariate subsets.

If the hypotheses are structured in such a way that many of the tested subsets are subsets of other sets, more powerful procedures can be applied that take advantage of this structure to gain power. Two methods are implemented in the `globaltest` package: the `inheritance` method for tree-structured hypotheses and the `focusLevel` method for general directed acyclic graphs. For simple multiple testing that does not use such structure, see `p.adjust`.

The `focusLevel` procedure makes use of the fact that some sets are subsets or supersets of each other, as specified by the user in the `offspring` and `ancestors` arguments. Viewing the subset and superset structure as a graph, the procedure starts testing at a `focus` level: a subset of the nodes of the graph. If the procedure finds significance at this focus level, it proceeds to find significant subsets and supersets of the focus level sets. Like Holm's procedure, the focus level procedure is valid regardless of the correlation structure between the test statistics.

The focus level method requires the choice of a "focus level" in the graph. The `findFocus` function is a utility function for automatically choosing a focus level. It chooses a collection of focus level sets in such a way that the number of tests to be done for each focus level node is at most `2^maxsize`. In practice this usually means that each focus level node has at most `maxsize` leaf nodes as offspring. Choosing focus level nodes with too many offspring nodes may result in excessively long computation times.

The `inheritance` method is an alternative method for calculating familywise error rate corrected p-values. Like the focus level method, `inheritance` also makes use of the structure of the tested sets to gain power. In this case, however, the graph is restricted to a tree, as can be obtained for example if the tested subsets are obtained from a hierarchical clustering. The inheritance procedure is used in the `covariates` function. Like Holm's method and the focus level method, the inheritance procedure makes no assumptions on the joint distribution of the test statistics.

The `leafNodes` function extracts the leaf nodes of the significant subgraph after a focus level procedure was performed. As this graph is defined by its leaf nodes, this is the most efficient summary of the test result. Only implemented for `gt.object` input.

The `draw` function draws the graph, displaying the significant nodes. It either draws the full graph with the non-significant nodes grayed out (`sign.only = TRUE`), or it draws only the subgraph corresponding to the significant nodes.

See the vignette for extensive applications.

Value

The function `multtest` returns an object of class `gt.object` with an appropriate column added to the test results matrix.

The `focusLevel` and `inheritance` functions returns a `gt.object` if a `gt.object` argument was given as input, otherwise it returns a matrix with a column of raw p-values and a column of corrected p-values.

The function `leafNodes` returns a `gt.object`.

`findFocus` returns a character vector.

Note

In the graph terminology of the focus level method, `ancestor` means superset, and `offspring` means subset.

The validity of the focus level procedure depends on certain assumptions on the null hypothesis that is tested for each set. See the paper by Goeman and Mansmann (cited below) for the precise assumptions. Similar assumptions are necessary for the Shaffer improvement of the inheritance procedure.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Livio Finos

References

The methods used by multtest:

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

Benjamini and Hochberg (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the Royal Statistical Society, Series B 57: 289-300.

Benjamini and Yekutieli (2001) The control of the false discovery rate in multiple testing under dependency. Annals of Statistics 29 (4) 1165-1188.

The focus level method:

Goeman and Mansmann (2008) Multiple testing on the directed acyclic graph of gene ontology. Bioinformatics 24 (4) 537-544.

The inheritance method:

Meinshausen (2008) Hierarchical testing of variable importance. Biometrika 95 (2), 265-278.

For references related to applications of the test, see the vignette GlobalTest.pdf included with this package.

See Also

The `gt` function. The `gt.object` function and useful functions associated with that object.

Many more examples in the vignette!

Examples

```
# Simple examples with random data here
# Real data examples in the Vignette

# Random data: covariates A,B,C are correlated with Y
set.seed(1)
Y <- rnorm(20)
X <- matrix(rnorm(200), 20, 10)
X[,1:3] <- X[,1:3] + Y
colnames(X) <- LETTERS[1:10]

# Some subsets of interest
my.sets1 <- list(abc = LETTERS[1:3], cde  = LETTERS[3:5],
                 fgh = LETTERS[6:8], hij = LETTERS[8:10])
res <- gt(Y, X, subsets = my.sets1)

# Simple multiple testing
p.adjust(res)
p.adjust(res, "BH")
```

```
# A whole structure of sets
my.sets2 <- as.list(LETTERS[1:10])
names(my.sets2) <- letters[1:10]
my.sets3 <- list(all = LETTERS[1:10])
my.sets <- c(my.sets2,my.sets1,my.sets3)

# Do the focus level procedure
# Choose a focus level by hand
my.focus <- c("abc","cde","fgh","hij")
# Or automated
my.focus <- findFocus(my.sets, maxsize = 8)
resF <- focusLevel(res, sets = my.sets, focus = my.focus)
leafNodes(resF, alpha = .1)

# Compare
p.adjust(resF, "holm")

# Focus level with a custom test
Ftest <- function(set) anova(lm(Y~X[,set]))[["Pr(>F)"]][1]
focusLevel(Ftest, sets=my.sets, focus=my.focus)

# analyze data using inheritance procedure
res <- gt(Y, X, subsets = list(colnames(X)))
# define clusters on the covariates X
hcl=hclust(dist(t(X)))
# Do inheritance procedure
resI=inheritance(res, sets = hcl)
resI
leafNodes(resI, alpha = .1)

# inheritance procedure with a custom test
inheritance(Ftest, sets = hcl, Shaffer=TRUE)
```

`GOstructure` *Class "GOstructure" (Deprecated)*

Description

The class `GOstructure` stores the Gene Ontology information that can be used by the `gtGO` function to find a GO subgraph significantly associated with a response variable. Objects of type `GOstructure` are typically made with a call to `makeGOstructure`.

Slots

`ids`: Object of class "character". Vector of GO identifiers.

`offspring`: A named list of vectors of GO identifiers. Lists all offspring terms of each term.

`ancestors`: A named list of vectors of GO identifiers. Lists all ancestor terms of each term.

`genesets`: A named list of vectors of gene identifiers. Lists the genes annotated to each GO id.

Methods

show (GOstructure): Summarizes the object.

length (GOstructure): Gives the number of GO ids in the object.

genesets (GOstructure): Extracts the genesets slot.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

See Also

`gtGO`, `makeGOstructure`.

`annotation.vandeVijver` *Some selected annotations for the breast cancer data from Rosetta*

Description

A named list of 64 vectors of probe identifiers appearing in the `vandeVijver`, linking the probes to annotation terms from the Gene Ontology (GO) database. All 64 GO terms are descendants of the GO term Cell Cycle (GO:0007049).

Usage

```
data(annotation.vandeVijver)
```

Format

A named list of vectors.

Note

The `vandeVijver` data set is deprecated. It was used for examples in an older version of the package, but is not used anymore. It will be removed from the package in a later version.

See Also

`vandeVijver`.

checkerboard — *Checkerboard plot for Global Test (Deprecated)*

Description

Produces a plot to visualize the test result produced by `globaltest`, by showing the association between pairs of samples.

Usage

```
checkerboard(gt, geneset, sort = TRUE, drawlabels = TRUE,
labelsize = 0.6, ...)
```

Arguments

`gt`	The output of a call to `globaltest`.
`geneset`	The name or number of the geneset to be plotted (only necessary if multiple genesets were tested)
`sort`	A logical flag to indicate whether the samples should be sorted by the clinical outcome to give a clearer picture.
`drawlabels`	Logical value to control drawing of the samplenames on the x- and y-axis of the plot.
`labelsize`	Relative size of the labels on the x- and y-axis. If it is `NULL` , the current value for `par("cex.axis")` is used
`...`	Any extra arguments will be forwarded to the plotting function.

Details

The checkerboard shows the pairs of samples which have high covariance in white and the pairs with low covariance in black. This can be used to visualize the data and to search for outlying arrays.

The left and bottom margins are adjusted to allow enough space for the longest samplename.

Value

A matrix giving the old and the new sample numbers.

Note

`checkerboard` has been deprecated. Please use `subjects` instead.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

References

J. J. Goeman, S. A. van de Geer, F. de Kort and J. C. van Houwelingen, 2004, *A global test for groups of genes: testing association with a clinical outcome*, *Bioinformatics* 20 (1) 93–99. See also the How To Globaltest.pdf included with this package.

See Also

globaltest, sampleplot, geneplot.

Examples

```
    # Breast cancer data (ExpressionSet) from the Netherlands Cancer
    # Institute with annotation:
    data(vandeVijver)
    data(annotation.vandeVijver)
    aPathway <- annotation.vandeVijver[1]

    gt <- globaltest(vandeVijver, "ESR1", aPathway)

    if (interactive()){
      checkerboard(gt[1])
    }
```

geneplot — *Gene Plot for Global Test (Deprecated)*

Description

Produces a plot to show the influence of individual genes on the test result produced by globaltest.

Usage

```
geneplot(gt, geneset, genesubset, scale = FALSE, drawlabels = TRUE,
  labelsize = 0.6, plot = TRUE, addlegend = TRUE, ...)
```

Arguments

`gt`	The output of a call to globaltest.
`geneset`	The name or number of the geneset to be plotted (only necessary if multiple genesets were tested).
`genesubset`	A vector of names or numbers of genes to be plotted (default: plot all genes)
`scale`	Logical: should the bars be scaled to unit standard deviation?
`drawlabels`	Logical value to control drawing of gene ids (rownames) on the x-axis of the plot.
`labelsize`	Relative size of the labels on the x-axis. If it is `NULL` , the current value for `par("cex.axis")` is used
`plot`	If `FALSE`: does not plot, but only returns a `gt.barplot` object.
`addlegend`	If `FALSE`: does not add a legend to the plot or to the `gt.barplot` object.
`...`	Any extra arguments will be forwarded to the plotting function.

Details

The geneplot shows a bar an a reference line for each gene. The bar shows the influence of each gene on the test result: the test statistic Q is the average of the bars of the genes. The reference line shows the expected influence if the gene was not associated with the outcome. The marks on the bars show the standard deviation of the bar under the null hypothesis. The color of the bar indicates positive or negative correlation of the gene with the clinical outcome, to distinguish between up and downregulation.

The bottom margin is adjusted to allow enough space for the longest gene id to draw under the axis.

Value

An object of type `gt.barplot`.

Note

`geneplot` has been deprecated. Please use `covariates` instead.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

References

For references, type: citation("globaltest"). See also the vignette GlobalTest.pdf included with this package.

See Also

`globaltest`, `sampleplot`.

Examples

```
# Breast cancer data (ExpressionSet) from the Netherlands Cancer
# Institute with annotation:
data(vandeVijver)
data(annotation.vandeVijver)

gt <- globaltest(vandeVijver, "StGallen", annotation.vandeVijver)

if (interactive()){
  geneplot(gt["cell cycle checkpoint"])
}

gp <- geneplot(gt["cell cycle checkpoint"], plot = FALSE)
if (interactive()){
  plot(sort(gp))
}
```

`Multiple testing on the GO graph`
Multiple testing on the GO graph (Deprecated)

Description

Three function to test (part of) the GO graph for association of the gene expression profile of GO terms with a response variable. Used together, these functions return multiplicity-adjusted p-values calculated using the Focus Level procedure that preserves the structure of the GO graph.

Usage

```
makeGOstructure(data, annotation, top, only.ids, ontology = c("BP", "CC", "MF"
getFocus(GOstructure, maxatoms = 10)
```

Arguments

`data`	The data set for which the `GOstructure` object is to be made. Can be either an `ExpressionSet` or a vector of probe indicators.
`annotation`	The name of the metaData annotation package to be used. If a metadata package is not available, the name of the organism package (e.g. `org.HS.eg.db` can be given instead, and the mapping from probe identifiers to entrez can be supplied with the `entrez` argument.
`top`	The node to be used as top node of the `GOstructure`. Defaults to one of "GO:0008150", "GO:0005575", "GO:0003674", depending on he ontology chosen. The `GOstructure` object will only contain offspring of the chosen top node.
`only.ids`	A vector of GO ids. If this is supplied, GO ids not appearing in this list will not appear in the `GOstructure` object.
`ontology`	The ontology to be used. One of Biological Process (BP), Cellular Component (CC) or Molecular Function (MF).
`entrez`	A vector of EntrezGene ids with the same length as the number of genes in `data`, (and in the same order) giving the mapping from probe identifiers to EntrezGene ids. If this argument is used, the `annotation` argument should give the name of the organism annotation package.
`unreliable`	Can be used to designate one or more of the GO evidence codes as unreliable. Value must be a vector containing one or more from "IC", "IDA", "IEA", "IEP", "IGI", "IMP", "IPI", "ISS", "NAS", "ND", "RCA", "TAS", "NR".
`GOstructure`	An object of class `GOstructure`.
`maxatoms`	A tuning parameter that governs the choice of the focus level. `getFocus` will choose those GO terms in the focus level in such a way that all offspring gene sets of each focus level term can be constructed as unions of no more than `maxatoms` atom gene sets. Default value of `maxatoms` is 10. Higher values quickly lead to slower computation. Lower values typically lead to reduced power.

Details

These functions should be used in the following order. First use `makeGOstructure` to make a GO graph tailored to a specific data set. Then `getFocus` can be used to choose a focus level. Finally `gtGO` performs the focus level procedure.

Value

The function returns a named vector of multiplicity-adjusted p-values. Adjusted p-values of GO terms not appearing in this vector are larger than the chosen value of `maxalpha`.

Note

`gtGO` cannot be used in combination with the permutation version of `globaltest`.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

References

For references, type: `citation("globaltest")`. See also the vignette GlobalTest.pdf included with this package.

See Also

Examples in the vignette! `globaltest`, `GOstructure`, `gt.multtest`.

`globaltest` *Global Test (Deprecated)*

Description

In microarray data, tests a (list of) group(s) of genes for significant association with a given clinical variable.

Usage

```
globaltest(X, Y, genesets, model,
    levels, d, event = 1, adjust,
    method = c("auto", "asymptotic", "permutations", "gamma"),
    nperm = 10^4, scaleX = TRUE, accuracy = 50, ...)
```

Arguments

`X`	Either a matrix of gene expression data, where columns correspond to samples and rows to genes or a Bioconductor `ExpressionSet`. The data should be properly normalized beforehand (and log- or otherwise transformed), but missing values are allowed (coded as `NA`). Gene and sample names can be included as the row and column names of `X`.

`Y`	A vector with the clinical outcome of interest, having one value for each sample. If `X` is an `ExpressionSet` it can also be the name of a covariate in the `phenoData` from the `ExpressionSet`, or a `formula` object using these names. If the clinical outcome is survival, `Y` should contain the survival times.
`genesets`	Either a vector or a list of vectors. Indicates the group(s) of genes to be tested. Each vector in `genesets` can be given in three formats. Either it can be a vector with 1 (`TRUE`) or 0 (`FALSE`) for each gene in `X`, with 1 indicating that the gene belongs to the group. Or it can be a vector containing the column numbers (in `X`) of the genes belonging to the group. Or it can be a subset of the rownames or `featureNames` for `X`.
`model`	Globaltest will try to determine the correct model from the input of `Y` and `d`. To override the automatic choice, use `model = "logistic"` for a two-valued outcome `Y`, `model = "linear"` for a continuous outcome and `model = "survival"` for a survival outcome.
`levels`	If `Y` is a factor (or a category in the PhenoData slot of `X`) and contains more than 2 levels: `levels` is a vector of levels of `Y` to test. If `levels` is length 2: test these 2 groups against each other. If levels is length 1: test that level against the others.
`d`	A vector or the name of a covariate in the `phenoData` from the `ExpressionSet` `X`, to indicate which samples experienced an event. Providing a value for `d` automatically sets `model = "survival"`
`event`	The value or values of `d` that indicates that there was an event.
`adjust`	Confounders or risk factors for which the test must be adjusted. Must be either a data frame or (if `X` is an `ExpressionSet`) the names of covariates in the `phenoData` from `X` or a `formula` object using these names. Default: no adjustment.
`method`	The method for calculation the p-value. Use `method = "asymptotic"` for the full asymptotic distribution of the test statistic; `method = "gamma"` for the gamma (= scaled chi-squared) approximation to that distribution and `method = "permutations"` for a permutation p-value. The default: `method = "auto"` chooses the permutations method if the number of possible permutations does not exceed 10,000 and the asymptotic otherwise. Note that `method = "gamma"` was the default option prior to version 4.0.0.
`nperm`	A number of permutations. This gives the (maximum) number of permutations to be used if `method = "permutations"` or `method = "auto"`.
`scaleX`	If true, rescales the expression matrix to get pleasant value for all test statistics. The expression matrix `X` is multiplied by a constant in such a way that the expected value EQ of the test statistic for the global test becomes exactly 10. This rescaling has no effect on the p-values.
`accuracy`	Numerical tuning parameter useable only with the asymptotic method and a non-survival response. Determines how much small eigenvalues of the `R` matrix are smoothed away to increase computation speed. Choose smaller values for quicker computations but conservative p-values; choose larger values for slower calculations but more accuracy.
`...`	Captures deprecated input for compatibility with older versions of globaltest.

Details

The Global Test tests whether a group of genes (of any size from one single gene to all genes on the array) is significantly associated with a clinical variable. The group could be for example a known pathway, an area on the genome or the set of all genes. The test investigates whether samples with similar clinical outcomes tend to have similar gene expression patterns. For a significant result it is not necessary that the genes in the group have similar expression patterns, only that many of them are correlated with the outcome.

Value

The function returns an object of class `gt.result`.

Note

`globaltest` has been deprecated. Please use `gt` instead.

The options globaltest options sampling and permutation have been replaced by separate functions from version 3.0. See `sampling` and `permutations`.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

References

For references, type: `citation("globaltest")`. See also the vignette GlobalTest.pdf included with this package.

See Also

Many more examples in the vignette! `geneplot`, `sampleplot`, `sampling`, `gt.multtest`, `permutations`, `checkerboard`, `regressionplot`.

Examples

```
    # Breast cancer data (ExpressionSet) from the Netherlands Cancer
    # Institute with annotation:
    data(vandeVijver)
    data(annotation.vandeVijver)

    # Many possible calls. See the vignette for more examples and explanation.
    globaltest(vandeVijver, "StGallen")
    globaltest(vandeVijver, "StGallen", annotation.vandeVijver)
    globaltest(vandeVijver, "Surv(TIMEsurvival, EVENTdeath)", annotation.vandeVijver)
    globaltest(vandeVijver, StGallen ~ Posnodes + StGallen, annotation.vandeVijver)
    globaltest(vandeVijver, "StGallen", method = "p")

    # Store the test result
    # See help(gt.result) for more options
    gt <- globaltest(vandeVijver, "StGallen", annotation.vandeVijver)
    gt[1:2]
    sort(gt)
    p.value(gt)

    # Also with simple vector/matrix input
    X <- matrix(rnorm(3000), 100, 30)   # random expression data
    Y <- 1:30                           # a response variable
    pathway <- 1:40                     # a pathway
```

```
globaltest(X, Y)
globaltest(X, Y, pathway)
```

`gt.barplot-class` *Class "gt.barplot" for results of the functions geneplot and sampleplot*

Description

The class gt.barplot stores the output of a call to `geneplot` or `sampleplot`.

Slots

`res`**:** Object of class "matrix". The plotted values, with a row for each gene or sample, and the following columns: `influence`: the height of the bar, `Einf`: expected influence under the null hypothesis, `sd.inf`: standard deviation of the influence under the null hypothesis, `up`: the colour of the plotted bar (1=green; 0=red)

`drawlabels`**:** Object of class "logical". Whether gene or sample labels should be printed at the x-axis.

`labelsize`**:** Object of class "numeric". The size of the gene/sample labels.

`legend`**:** The meaning of the colours of the bars, to be used in the legend.

`colour`**:** The numbers of the colours used in the plot.

`colourCode`**:** The meaning of the colours used (shorter version of `legend`.

Methods

show (gt.barplot): Summarizes plot in numbers.

plot (gt.barplot): Redraws the plot. Possible extra arguments: `labelsize` and `drawlabels`.

"[" (gt.barplot): Extracts a subset of the genes or samples.

result (gt.barplot): Extracts the results matrix as a `data.frame`.

length (gt.barplot): The number of bars (genes or samples) in the object.

scale (gt.barplot): Scales the bars to unit standard deviation.

z.score (gt.barplot): Calculates the z.scores for each bar: (influence - expected inf) / sd(inf).

sort (gt.barplot): Sorts the bars to a decreasing z.score.

names (gt.barplot): The names of the genes or samples.

names<- (gt.result): Changes the names of the genes or samples.

Note

`gt.barplot` has been deprecated: please use use `covariates` or `subjects` instead.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

See Also

`sampleplot`, `geneplot`, `globaltest`.

gt.result-class *Class "gt.result" for results of the function globaltest (Deprecated)*

Description

The class gt.result is the output of a call to `globaltest` and the input of various plotting functions to visualize the test result.

Slots

`res`**:** Object of class "matrix". Test results summary.

`eX`**:** Object of class "matrix". The transformed data matrix.

`genesets`**:** A list of vectors indicating the tested genes.

`IminH`**:** Object of class "matrix". Needed when drawing the various diagnostic plots.

`fit`**:** Object of class "list". This list contains one element which is of class "lm", "glm", "coxph" or "coxph.null". This latter object contains the fitted model of the null hypothesis.

`method`**:** Object of class "numeric". Indicates the method used for p-value calculation.

`PermQs`**:** Object of class "matrix". Stores the permuted test statistics generated by a call to `permutations`.

`samplingZs`**:** Object of class "list". Stores the standardized test statistics of random gene sets generated by a call to `sampling`.

Methods

show (gt.result): Summarizes the test result.

"[" (gt.result): Extracts results of one or more pathways if multiple pathways were tested.

length (gt.result): The number of pathways tested.

size (gt.result): Extracts a vector with the number of genes tested for each pathway.

p.value (gt.result): Extracts the p-values.

z.score (gt.result): Extracts z-score (Q - EQ) / sd(Q).

sort (gt.result): Sorts the pathways to increasing p-values.

result (gt.result): Extracts the results matrix.

names (gt.result): Extracts the pathway names.

names<- (gt.result): Changes the pathway names.

combine (gt.result): Combines two gt.result objects to one, provided the data and model match.

fit (gt.result): Extracts the fitted (adjust)model.

hist (gt.result): Produces a histogram to visualize the permutations generated by `permutations`.

geneplot (gt.result): The `geneplot` produces a plot to show the influence of individual genes on the test result produced by `globaltest`.

sampleplot (gt.result): The `sampleplot` produces a plot to show the influence of individual samples on the test result produced by `globaltest`.

permutations (gt.result): The function `permutations` recalculates the p-values using permutations of the outcome Y.

sampling (gt.result): The function `sampling` compares the p-values with p-values of randomly generated pathways.

checkerboard (gt.result): Produces a plot to visualize the test result produced by `globaltest` by showing the association between pairs of samples.

regressionplot (gt.result): Produces a plot which can be used to visualize the effect of specific samples on the test result produced by `globaltest`.

Note

`gt.result` has been deprecated: please use use `gt.object` instead.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

See Also

`globaltest`, `sampleplot`, `geneplot`, `permutations`.

`gt.multtest` *Correct globaltest results for multiple testing (Deprecated)*

Description

(DEPRECATED: please use use `p.adjust` instead). Corrects the raw p-values resulting from a call to `globaltest` for multiple testing, using either Benjamini and Hochberg's False Discovery Rate or Holm's procedure for controlling the Family-Wise Error Rate.

Usage

```
gt.multtest(gt, proc = c("FDR", "FWER"))
```

Arguments

`gt`	The output of a call to `globaltest`.
`proc`	The procedure to be used. Either "FDR" for Benjamini and Hochberg's (1995) False Discovery Rate-controlling procedure or "FWER" for Holm's (1979) Family-Wise Error Rate controlling procedure.

Details

This function is completely based on the `mt.rawp2adjp` function from the `multtest` package.

Value

An object of class `gt.result`.

Note

`gt.multtest` has been deprecated. Please use `p.adjust` instead.

This function must be called *prior* to any selection of significant genes.

Author(s)

Jelle Goeman: <j.j.goeman@lumc.nl>; Jan Oosting

References

For references, type: citation("globaltest"). See also the vignette GlobalTest.pdf included with this package.

See Also

globaltest, gtGO.

Examples

```
    # Breast cancer data (ExpressionSet) from the Netherlands Cancer
    # Institute with annotation:
    data(vandeVijver)
    data(annotation.vandeVijver)

    gt <- globaltest(vandeVijver, "StGallen", annotation.vandeVijver)
    sort(gt.multtest(gt))
```

---

permutations   *Permutations version of the Global Test (Deprecated)*

Description

Produces permutations of the globaltest test statistic and uses these to recalculate the p-value.

Usage

```
permutations(gt, geneset, nperm = 10^4)
```

Arguments

gt	The output of a call to globaltest.
geneset	The name or number of the geneset(s) to be used (only necessary if multiple genesets were tested).
nperm	The number of permutations to be used.

Details

A permutation p-value is calculated by comparing the value of the test statistic using permutations of the clinical outcome to the value of the test statistic for the true clinical outcome. If the number of possible permutations is smaller than the value of nperm, all possible permutations are used, otherwise nperm random permutations. The permuted test statistics are stored for later visualisation using hist.

Value

An object of class gt.result.

Note

`permutations` has been deprecated: please use use `gt` instead.

Permutations does not work if the adjusted version of globaltest was used.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

References

For references, type: citation("globaltest"). See also the vignette GlobalTest.pdf included with this package.

See Also

`globaltest`, `sampleplot`, `geneplot`.

Examples

```
    # Breast cancer data (ExpressionSet) from the Netherlands Cancer
    # Institute with annotation:
    data(vandeVijver)
    data(annotation.vandeVijver)
    aPathway <- annotation.vandeVijver[1]

    gt <- globaltest(vandeVijver, "StGallen", aPathway)
    perm.gt <- permutations(gt)

    if (interactive()) {
      hist(perm.gt)
    }
```

`regressionplot` *Regression Plot for Global Test (Deprecated)*

Description

Produces a plot which can be used to visualize the effect of specific samples on the test result produced by `globaltest`.

Usage

```
regressionplot(gt, geneset, sampleid, ...)
```

Arguments

`gt`	The output of a call to `globaltest`.
`geneset`	The name or number of the geneset to be plotted (only necessary if multiple genesets were tested).
`sampleid`	An optional vector of names or numbers of the samples of interest.
`...`	Any extra arguments will be forwarded to the plotting function.

Details

The regressionplot plots, for all pairs of samples, the covariance between the expression patterns against the covariance between their clinical outcomes. Each point in the plot therefore represents a pair of samples. A regression line is fitted through the samples, which visualizes the test result of the function `globaltest`. A steeply increasing slope indicates a high (possibly significant) value of the test statistic.

An optional argument `sampleid` can be supplied, giving sample numbers of possibly outlying arrays. In this case, all pairs of arrays involving one of the arrays in `sampleid` is marked as a red cross, while the other pairs are marked as a blue dot. The blue line which is fitted through all points can now be compared to a red dotted line which is fitted though only the red crosses.

Value

`NULL` (no output).

Note

`regressionplot` has been deprecated. Please use `subjects` instead.

Regressionplot does not work if the adjusted version of globaltest was used.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

References

J. J. Goeman, S. A. van de Geer, F. de Kort and J. C. van Houwelingen, 2004, *A global test for groups of genes: testing association with a clinical outcome*, *Bioinformatics* 20 (1) 93–99. See also the How To Globaltest.pdf included with this package.

See Also

`globaltest`, `sampleplot`, `geneplot`.

Examples

```
    # Breast cancer data (ExpressionSet) from the Netherlands Cancer
    # Institute with annotation:
    data(vandeVijver)
    data(annotation.vandeVijver)

    gt <- globaltest(vandeVijver, "ESR1", annotation.vandeVijver)

    if (interactive()) {
      regressionplot(gt[1])
      regressionplot(gt[1], sampleid = "122")
    }
```

`sampleplot` *Sample Plot for Global Test (Deprecated)*

Description

Produces a plot to show the influence of individual samples on the test result produced by `globaltest`.

Usage

```
sampleplot(gt, geneset, samplesubset, scale = TRUE, drawlabels = TRUE,
  labelsize = 0.6, plot = TRUE, addlegend = TRUE, ...)
```

Arguments

`gt`	The output of a call to `globaltest`.
`geneset`	The name or number of the geneset to be plotted (only necessary if multiple genesets were tested).
`samplesubset`	A vector of names or numbers of samples to be plotted. Default: plot all samples
`scale`	Logical: should the bars be scaled to unit standard deviation?
`drawlabels`	Logical value to control drawing of the samplenames on the x-axis of the plot.
`labelsize`	Relative size of the labels on the x-axis. If it is `NULL` , the current value for `par("cex.axis")` is used
`plot`	If `FALSE`: does not plot, but only returns a `gt.barplot` object.
`addlegend`	If `FALSE`: does not add a legend to the plot or to the `gt.barplot` object.
`...`	Any extra arguments will be forwarded to the plotting function.

Details

The sampleplot shows a bar and a reference line for each sample. The bar shows the influence of each gene on the test statistic. Samples with a positive influence carry evidence against the null hypothesis (in favour of a significant p-value), because they are are similar in expression profile to samples with a similar clinical outcome. Samples with a negative influence bar supply evidence in favour of the null hypothesis and of a non-significant p-value: they are relatively similar in expression profile to samples with a different clinical outcome.

The influence varies around zero if the tested geneset is not associated with the outcome. Marks on the bars show the standarddeviation of the influence under the null hypothesis for those samples which are more than one standard deviation away from zero.

The color of the bar indicates the sign of the residual of Y. In a logistic model the coloring this distinguishes the original groups.

The bottom margin is adjusted to allow enough space for the longest samplename to draw under the axis.

Value

An object of type `gt.barplot`.

Note

`sampleplot` has been deprecated. Please use `subjects` instead.

Author(s)

Jelle Goeman: <j.j.goeman@lumc.nl>; Jan Oosting

See Also

globaltest, geneplot, regressionplot, checkerboard.

Examples

```
# Breast cancer data (ExpressionSet) from the Netherlands Cancer
# Institute with annotation:
data(vandeVijver)
data(annotation.vandeVijver)

gt <- globaltest(vandeVijver, "StGallen", annotation.vandeVijver)

if (interactive()){
  sampleplot(gt[1])
}

sp <- sampleplot(gt[1], plot = FALSE)
if (interactive()){
  plot(sort(sp))
}
```

---

sampling — *Sampling random 'pathways' for the Global Test*

Description

(DEPRECATED: please use use comparative instead). For every pathway in a result of globaltest, calculates how many randomly drawn groups of genes of the same size have a smaller or equal p-value.

Usage

```
sampling(gt, geneset, ndraws = 10^3)
```

Arguments

gt	The output of a call to globaltest.
geneset	The name or number of the geneset(s) to be used (only necessary if multiple genesets were tested).
ndraws	The number of random pathways to be used.

Details

For every pathway in gt[geneset], a number ndraws random 'pathways' is selected by randomly sampling sets of genes of the same size as the tested pathway. A 'comparative p-value' is calculated by counting what proportion of the random pathways has a larger standardized test statistic (Q - EQ) / sd(Q) than the tested pathway.

Value

An object of class `gt.result`.

Note

`sampling` has been deprecated. Please use `comparative` instead.

The function `sampling` cannot be applied to a `gt.result` object resulting from a call to `permutations`.

Author(s)

Jelle Goeman: `<j.j.goeman@lumc.nl>`; Jan Oosting

References

Jelle J. Goeman, Jan Oosting, Anne-Marie Cleton-Jansen, Jakob, K. Anninga, Hans C. van Houwelingen (2005) Testing association of a pathway with survival. Bioinformatics 21, 1950-1957. See also the vignette Globaltest.pdf that comes with this package.

See Also

`globaltest`, `permutations`, `sampleplot`, `geneplot`.

Examples

```
    # Breast cancer data (ExpressionSet) from the Netherlands Cancer
    # Institute with annotation:
    data(vandeVijver)
    data(annotation.vandeVijver)

    gt <- globaltest(vandeVijver, "StGallen", annotation.vandeVijver)

    sampling(gt, ndraws=100)
```

`vandeVijver` *A subset of the Breast cancer data from Rosetta Inpharmatics LLC and*

Description

An `ExpressionSet` containing expression data of 100 breast cancer patients on 407 selected genes plus the following phenotype data.

SampleID Sample ID given by NKI for 295 samples in cohort study (NEJM 347, 01999, 2002)

FirstSeriesID Sample ID in first series given by NKI for 78 samples in our first study (Nature 415, p530, 2002)

Posnodes Lymph node status from pathology report

EVENTmeta Distant metastasis at any time during follow-up, 0 = no, 1 = yes

EVENTdeath Death, 0 = no, 1 = yes

TIMEsurvival Total survival interval in years between first date of treatment and last date of follow-up (if EVENTdeath = 0) or date of death (if EVENTdeath = 1)

TIMErecurrence Disease free interval in years between first date of treatment and last date of follow-up (if all 'EVENTs' are 0) or date of first EVENT (if one ore more 'EVENTs' are 1, i.e., local recurrence, loco-regional recurrence, second primary tumor or distant metastasis)

TIMEmeta Metastasis free interval in years between first date of treatment and date of diagnosis of distant metastasis (only if EVENTmeta = 1) (note that interval relates to metastasis at any time during follow-up and not just as first event)

ESR1 Estrogen receptor alpha expression measurement from microarray (0 = no, 1 = yes)

NIH Patient characteristics according to NIH criteria, 1 = low risk, 0 = high risk

St Gallen Patient characteristics according to St. Gallen criteria, 1 = low risk, 0 = intermediate/high risk

Usage

```
data(vandeVijver)
```

Format

An `ExpressionSet`. See the Biobase package

Note

The `vandeVijver` data set is deprecated. It was used for examples in an older version of the package, but is not used anymore. It will be removed from the package in a later version.

References

M.J. van de Vijver, Y.D. He, L.J. van 't Veer, H. Dai, A.A.M. Hart, D.W. Voskuil, G.J. Schreiber, J.L. Peterse, C. Roberts, M.J. Marton, M. Parrish, D. Atsma, A. Witteveen, A. Glas, L. Delahaye, T. van der Velde, H. Bartelink, S. Rodenhuis, E.T. Rutgers, S.H. Friend, and R. Bernards (2002). A gene-expression signature as a predictor of survival in breast cancer. New England Journal of Medicine 347 (25), 1999–2009.

See Also

`annotation.vandeVijver`.

Index