Package 'globaltest'

November 12, 2026

Version 5.66.0

Date 2023-03-30

Title Testing Groups of Covariates/Features for Association with a Response Variable, with Applications to Gene Set Testing

Author
Jelle Goeman and Jan Oosting, with contributions by Livio Finos, Aldo Solari, Dominic Edelmann

Maintainer Jelle Goeman <j.j.goeman@lumc.nl>

Depends methods, survival

Imports Biobase, AnnotationDbi, annotate, graphics

Suggests vsn, golubEsets, KEGGREST, hu6800.db, Rgraphviz, GO.db, lungExpression, org.Hs.eg.db, GSEABase, penalized, gss, MASS, boot, rpart, mstate

Description The global test tests groups of covariates (or features) for association with a response variable. This package implements the test with diagnostic plots and multiple testing utilities, along with several functions to facilitate the use of this test for gene set testing of GO and KEGG terms.

License GPL (>= 2)

biocViews Microarray, OneChannel, Bioinformatics, DifferentialExpression, GO, Pathways

Collate gt-object.R mlogit.R comparative.R diagnosticplots.R focuslevel.R generalizedFdistribution.R genesettesting.R gt.R inheritance.R models.R options.R utilities.R goodnessoffit.R

LazyLoad yes

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

git_branch RELEASE_3_23

git_last_commit 8ca8e77

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Contents

Comparative proportions . . . 2
Diagnostic plots for globaltest . . . 3
gt . . . 7
gt gene set testing methods . . . 11
gt goodness of fit methods . . . 14
gt.object class . . . 18
gt.options . . . 21
mlogit . . . 22
Multiple testing methods of the globaltest package . . . 23

Index **29**

`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, plot = TRUE, colors, alias, help.lines = FALSE,
           cex.labels = 0.6, ylim, 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, ylim, 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.
plot	If FALSE, suppress all plotting.
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.
ylim	Override for the y axis limits of the barplot.

`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])

# Extract data from the plot
out <- covariates(res)
result(out)
extract(out)
```

---

gt *Global Test*

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. When using the dot in a formula, make sure to avoid the

half formula (~., but repeat the response on the left hand side. 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. When given as a half formula object, one may also specifiy strata (e.g. ~strata(a)+b)

).

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 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://biosemantics.org/index.php/software/weighted-global-test.
concept2name	The name of the file containing a mapping between Anni concepts and entrez identifiers. Useable files can be downloaded from http://biosemantics.org/index.php/software/weighted-global-test.
entrez2concept	The name of the file containing a mapping between Anni concept numbers and names. Useable files can be downloaded from http://biosemantics.org/index.php/software/weighted-global-test.
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://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/downloads.jsp`, and to preprocess that file using the `getBroadSets` function. `gtConcept` requires files that can be downloaded from `http://biosemantics.org/index.php/software/weighted-global-test`.

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, `gtKS` uses Kernel Smoothers and `gtLI` uses Linear Interactions. The other functions are for external use in combination with `gt`.

Usage

```
gtPS(response, null, data,
      model = c("linear", "logistic", "cox", "poisson", "multinomial"),
      ..., covs, bdeg = 3, nint= 10, pord = 2, interact = FALSE, robust = FALSE,
      termlabels = FALSE, returnZ = FALSE)

gtKS(response, null, data,
      model = c("linear", "logistic", "cox", "poisson", "multinomial"),
      ..., covs, quant = .25, metric = c("euclidean", "pearson"),
      kernel=c("uniform", "exponential", "triangular", "neighbours", "gauss"),
      robust = FALSE, scale = TRUE, termlabels = FALSE, returnZ = FALSE)

gtLI(response, null, data, ..., covs, iorder=2, termlabels = FALSE, standardize = FALSE)

bbase(x, bdeg, nint)

btensor(xs, bdeg, nint, pord, returnU=FALSE)

reparamZ(Z, pord, K=NULL, tol = 1e-10, returnU=FALSE)

reweighZ(Z, null.fit)

sterms(object, ...)
```

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, with default bdeg=3.
nint	A vector or a list of vectors which specifies the number of intervals determined by equally-spaced knots, with default nint=10.
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, with default pord=2.
interact	TRUE to consider a multidimensional smooth function of covs.
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).
returnZ	TRUE gives back the alternative design matrix used in the test.
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 center and scale the covariates before computing the distance.
iorder	Order of the linear interactions, e.g. second order interactions, third order etc., with default iorder=2.
standardize	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.
x	A numeric vector of values at which to evaluate the B-spline basis.
xs	A matrix or dataframe where the columns correspond to covariates values.
returnU	codeTRUE gives back the nonpenalized part.
Z	Alternative design matrix.
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.
null.fit	Fitted null model.
object	A gt.object from gtPS, gtKS or gtLI.

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 that accomodate 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 constructed 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. If `interact=TRUE`, the alternative is given by a multidimensional smooth function of `covs`, which is represented by a tensor product of marginal B-splines bases and Kronecker sum of the marginal penalties, e.g. `alternative= ~ s(X1,X2)` when `covs=c("X1","X2")` and `interact=TRUE`.

By using `gtKS` 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 (to be implemented).

By using `gtLI`, the alternative is given by all the possible ith-order linear interactions between `covs`, e.g. `alternative= ~ X1:X2 + X1:X3 + X2:X3` when `covs=c("X1","X2","X3")` and `iorder=2`.

The remaining functions are meant for constructing the alternative design matrix that will be used in the `alternative` argument of the `gt` function. `bbase` constructs the B-spline basis for the covariate `x`. This function is based on the functions provided by Eilers and Marx (1996). `btensor` builts a tensor product of B-splines for the covariates `xs`, which is reparameterized according with a Kroneker sum of penalties. `reparamZ` reparameterizes the alternative design matrix (e.g. a spline basis `B`) according with the order of differences `pord` or via the spectral decomposition of a roughness matrix `K`. When several smooth terms are to be combined, `reweighZ` assigns equal weight to each component term.

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.

Methods

sterms (gt.object): Prints the smooth terms specified by `gtPS`, `gtKS` or `gtLI`.

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
   set.seed(0)
   X1<-runif(50)
   s1 <- function(x) exp(2 * x)
   e <- rnorm(50)
   Y <-  s1(X1) + e

   ### gtPS
   res<-gtPS(Y~X1)
   res@result
   sterms(res)

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

   # formula input and termlabels
   gtPS(Y~exp(2*X1),data=rdata)
   gtPS(Y~exp(2*X1),covs="exp(2 * X1)",data=rdata)
   sterms(gtPS(Y~exp(2*X1),data=rdata,termlabels=TRUE))

   # P-splines arguments
   gtPS(Y~X1, bdeg=3, nint=list(a=10, b=30), pord=0)
   gtPS(Y~X1, bdeg=3, nint=list(a=10, b=30), pord=0, 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, covs="X2")
   sterms(gtPS(Y~X1+X2, nint=list(a=c(10,30), b=20)))
```

```
# 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

# Tensor product of P-splines
res<-gtPS(Y~X1*X2, interact=TRUE)
res@result
sterms(res)

### gtKS
res<-gtKS(Y~X1*X2)
res@result
sterms(res)
gtKS(Y~X1*X2, quant=seq(.05,.95,.05), robust=TRUE)

### gtLI
library(MASS)
data(Boston)
gtLI(medv~., data=Boston, standardize=TRUE)
```

`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.

extract (gt.object): Extracts the results matrix for the leaf nodes after a call to `link{covariates}`, with information on direction of association.

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.

sterms (gt.object): Prints the smooth terms specified by `gtPS`, `gtKS` or `gtLI`.

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])
extract(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: `TRUE`). 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 applys 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 leaf 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)
```

Index

∗ **goodness-of-fit**
    gt goodness of fit methods, 14
∗ **htest**
    Comparative proportions, 2
    Diagnostic plots for globaltest, 3
    gt, 7
    gt gene set testing methods, 11
    gt.options, 21
    Multiple testing methods of the globaltest package, 23
∗ **methods**
    gt.object class, 18
∗ **nonlinear**
    mlogit, 22
[,gt.object,ANY,ANY,ANY-method (gt.object class), 18
[,gt.object-method (gt.object class), 18
[[,gt.object-method (gt.object class), 18

alias, *9*
alias,gt.object-method (gt.object class), 18
alias<- (gt.object class), 18
alias<-,gt.object-method (gt.object class), 18

bbase (gt goodness of fit methods), 14
btensor (gt goodness of fit methods), 14

coefficients,mlogit-method (mlogit), 22
comparative, *18*, *20*
comparative (Comparative proportions), 2
Comparative proportions, 2
covariates, *6*, *10*, *18–20*, *25*
covariates (Diagnostic plots for globaltest), 3
coxph, *14*

dendrogram, *24*
Diagnostic plots, *9*
Diagnostic plots for globaltest, 3
diagnostics (Diagnostic plots for globaltest), 3
draw (Multiple testing methods of the globaltest package), 23

ExpressionSet, *8*, *12*
extract (gt.object class), 18
extract,gt.object-method (gt.object class), 18

factor, *8*
features (Diagnostic plots for globaltest), 3
findFocus, *12*
findFocus (Multiple testing methods of the globaltest package), 23
fitted.values,mlogit-method (mlogit), 22
focusLevel, *12*
focusLevel (Multiple testing methods of the globaltest package), 23
formula, *7–9*, *14*, *22*
function, *24*

gene set testing, *9*
getBroadSets, *12*, *13*
glm, *14*, *22*, *23*
glm.control, *22*
globaltest (gt), 7
gt, *3–5*, 7, *11–13*, *15*, *17–20*, *22*, *24*, *27*
gt gene set testing methods, 11
gt goodness of fit methods, 14
gt.object, *2–4*, *6*, *9*, *10*, *13*, *15–17*, *24–27*
gt.object (gt.object class), 18
gt.object class, 18
gt.object-class (gt.object class), 18
gt.options, *5*, *9*, 21
gt.options()$trace, *24*

gtBroad (gt gene set testing methods), 11
gtConcept (gt gene set testing methods), 11
gtGO (gt gene set testing methods), 11
gtKEGG (gt gene set testing methods), 11
gtKEGGREST (gt gene set testing methods), 11
gtKS, *15, 16*, *20*
gtKS (gt goodness of fit methods), 14
gtLI, *15, 16*, *20*
gtLI (gt goodness of fit methods), 14
gtPS, *15, 16*, *20*
gtPS (gt goodness of fit methods), 14

hclust, *4*, *24*
hist,gt.object-method (gt.object class), 18

inheritance (Multiple testing methods of the globaltest package), 23
interactive, *2*

leafNodes (Multiple testing methods of the globaltest package), 23
length,gt.object-method (gt.object class), 18
list, *24*
lm, *14*

mlogit, 22
mlogit-class (mlogit), 22
model.matrix,gt.object-method (gt.object class), 18
multinom, *23*
Multiple testing methods of the globaltest package, 23

names,gt.object-method (gt.object class), 18
names<-,gt.object-method (gt.object class), 18

p.adjust, *18*, *20*, *25*
p.adjust,gt.object-method (gt.object class), 18
p.value (gt.object class), 18
p.value,gt.object-method (gt.object class), 18

reparamZ (gt goodness of fit methods), 14
residuals,mlogit-method (mlogit), 22
result (gt.object class), 18
result,gt.object-method (gt.object class), 18
reweighZ (gt goodness of fit methods), 14
rgb, *4*

show,gt.object-method (gt.object class), 18
show,mlogit-method (mlogit), 22
size (gt.object class), 18
size,gt.object-method (gt.object class), 18
sort,gt.object-method (gt.object class), 18
sterms (gt goodness of fit methods), 14
sterms,gt.object-method (gt.object class), 18
subjects, *6*, *10*, *18–20*
subjects (Diagnostic plots for globaltest), 3
subsets (gt.object class), 18
subsets,gt.object-method (gt.object class), 18
summary,gt.object-method (gt.object class), 18
summary,mlogit-method (mlogit), 22

weights, *25*
weights,gt.object-method (gt.object class), 18

z.score (gt.object class), 18
z.score,gt.object-method (gt.object class), 18