safe

March 24, 2012

`SAFE-class` *Class SAFE*

Description

The class SAFE is the output from the function `safe`. It is also the input to the plotting function `safeplot`.

Slots

`local`: Object of class `"character"` describing the local statistic used.

`local.stat`: Object of class `"numeric"` containing the (unsorted) observed local statistics for genes.

`local.pval`: Object of class `"numeric"` containing the (unsorted) empirical p-values for genes

`global`: Object of class `"character"` describing the local statistic used.

`global.stat`: Object of class `"numeric"` containing the (unsorted) observed global statistics for categories.

`global.pval`: Object of class `"numeric"` containing the (unsorted) empirical p-values for categories.

`error`: Object of class `"character"` describing the method used to estimate error rates across multiple comparisons.

`global.error`: Object of class `"numeric"` containing the (unsorted) error rates associated with the p-values for categories. If not computed, it will be set to `NA`.

`C.mat`: Object of class `"matrix"` containing the category assignments. Each row corresponds to a gene, and each column a category.

`alpha`: Object of class `"numeric"` containing the alpha level for significance of a category.

`method`: Object of class `"character"` describing the resampling method used in `safe`.

Methods

show (gt.result): Summarizes the test results of significant categories.

[(gt.result): Returns a SAFE object for categories indicated by integer of character strings.

safeplot (gt.result): The `safeplot` produces a plot of the relative association of expression in a category of genes relative to their complement.

Author(s)

William T Barry: `<bill.barry@duke.edu>`

See Also

`safe`, `safeplot`.

`gene.results` *Gene-specific results from SAFE*

Description

Prints gene-specific local statistics and resampling-based p-values for every probeset in the gene category of interest. Probesets are ordered by the degree and direction of differential expression.

Usage

```
gene.results(object = NULL, cat.name = NULL, error = "none", print.it = TRUE)
```

Arguments

`object`	Object of class `SAFE`.
`cat.name`	Name of the category to be plotted. If omitted, the most significant category is plotted.
`error`	Specifies a non-resampling based method for adjusting the empirical p-values. A Bonferroni, ("FWER.Bonf"), Holm's step-up ("FWER.Holm"), and Benjamini-Hochberg step down ("FDR.BH") adjustment can be selected. By default ("none") no error rates are computed.
`print.it`	Logical determining whether results are printed to screen or returned as a list of results for up- and down-regulated genes.

Author(s)

William T. Barry: `<bill.barry@duke.edu>`

References

W. T. Barry, A. B. Nobel and F.A. Wright, 2005, *Significance Analysis of functional categories in gene expression studies: a structured permutation approach*, *Bioinformatics***21**(9) 1943–1949.

See also the vignette included with this package.

See Also

`safe`.

`getCmatrix` *Generation of a C matrix*

Description

This function will convert a list, vector or file of gene annotation into a C matrix. Size constraints, and present/absent calls can be set to filter categories and genes accordingly.

Usage

```
getCmatrix(keyword.list = NULL, gene.list = NULL, vector = NULL, file = NULL,
           delimiter = ",", present.genes = NULL, GO.ont = NULL, min.size = 0,
           max.size = Inf, as.matrix = FALSE, ...)
```

Usage

```
getCmatrix(keyword.list, present.genes=, GO.ont=)
getCmatrix(gene.list, present.genes=, min.size=, max.size=)
getCmatrix(vector=, delimiter=, as.matrix=)
getCmatrix(file=, delimiter=, ...)
```

Arguments

`keyword.list`	A list containing character vectors for each keyword that specify the gene members.
`gene.list`	A list containing character vectors for each gene that specify the functional categories it belongs to.
`vector`	A character vector of gene annotation with a specified delimiter between category keywords.
`file`	A file containing the character vector of gene annotation.
`delimiter`	Delimiter used between category keywords when provided as a vector or file.
`present.genes`	An optional vector used to filter genes in the C matrix. Can be provided as an unordered character vector of gene names that match `names(list)`, or as an ordered vector of presence (`1`) and absence (`0`) calls.
`GO.ont`	"CC","BP",or "MF" specify the ontology to limit categories to.
`min.size`	Optional minimum category size to be considered.
`max.size`	Optional maximum category size to be considered.
`as.matrix`	Optional argument to specify a matrix is returned rather than a matrix.csr.
`...`	Any extra arguments will be forwarded to the read.table function when category assignments are given as a file.

Value

`C.mat.csr`	If `as.matrix=F` a sparse matrix is returned with the rows corresponding to the genes and columns are categories
`row.names`	Character vector of gene names
`col.names`	Character vector of category names

Author(s)

William T. Barry: `<bill.barry@duke.edu>`

References

W. T. Barry, A. B. Nobel and F.A. Wright, 2005, *Significance Analysis of functional categories in gene expression studies: a structured permutation approach*, *Bioinformatics* **21**(9) 1943-9.

See also the vignette included with this package.

See Also

`safe`, `safeplot`, `getPImatrix`.

Examples

```
## A simple illustration
anno <- c("Keyword1","Keyword2;Keyword3","",
          "Keyword3;Keyword1","Keyword3")
names(anno) <- paste("Gene",1:5)

getCmatrix(vector = anno, delimiter = ";",
           as.matrix=TRUE)
```

`safe` *Significance Analysis of Function and Expression*

Description

Performs a significance analysis of function and expression (SAFE) for a given gene expression experiment and a given set of functional categories. SAFE is a two-stage permutation-based method that can be applied to a 2-sample, multi-class, simple linear regression, and other linear models. Other experimental designs can also be accommodated through user-defined functions.

Usage

```
safe(X.mat, y.vec, C.mat = NULL, platform = NULL, annotate = NULL, Pi.mat = NULL
     local = "default", global = "Wilcoxon", args.local = NULL,
     args.global = list(one.sided = FALSE), error = "none", alpha = NA,
     method = "permutation", min.size = 2, max.size = Inf, ...)
```

Arguments

`X.mat`	A matrix or data.frame of expression data; each row corresponds to a gene and each column to a sample. Data can also be given as the Bioconductor class `ExpressionSet`. Data should be properly normalized and may not contain missing values.
`y.vec`	a numeric, integer or character vector of length `ncol(X.mat)` containing the response of interest. If `X.mat` is an `ExpressionSet`, `y.vec` can also be the name or column number of a covariate in the `phenoData` slot. For examples of the acceptable forms `y.vec` can take, see the vignette.

`C.mat`	A matrix or data.frame containing the gene category assignments. Each column represents a category and should be named accordingly. For each column, values of 1 (`TRUE`) and 0 (`FALSE`) indicate whether the genes in the corresponding rows of `X.mat` are contained in the category. This can also be a list containing a sparse matrix and dimnames as created by `getCmatrix`
`platform`	If `C.mat` is unspecified, a character string of a Bioconductor annotation package can be used to build gene categories. See vignette for details and examples.
`annotate`	If `C.mat` is unspecified, a character string to specify the type of gene categories to build from annotation packages. "GO.MF", "GO.BP", "GO.CC", and "GO.ALL" (default) specify one or all Gene Ontologies. "KEGG" specifies pathways, and "PFAM" homologous families from the respective sources.
`Pi.mat`	Either a matrix or data.frame containing the permutations, or an integer. See `getPImatrix` for the acceptable form of a matrix or data.frame. If `Pi.mat` is an integer, then `safe` will automatically generate as many random permutations of `X.mat`.
`local`	Specifies the gene-specific statistic from the following options: "t.Student", "t.Welch" and "t.SAM" for 2-sample designs, "f.ANOVA" for 1-way ANOVAs, "t.LM" for simple linear regressions, and "z.COXPH" for a Cox proportional hazards survival model. "default" will choose between "t.Student" and "f.ANOVA", based on the form of `y.vec`. User-defined local statistics can also be used; details are provided in the vignette.
`global`	Specifies the global statistic for a gene categories. By default, the Wilcoxon rank sum ("Wilcoxon") is used. Else, a Fisher's Exact test statistic ("Fisher") based on the hypergeometric dist'n, a chi-squared type Pearson's test ("Pearson") or t-test of average difference ("AveDiff") is available. User-defined global statistics can also be implemented.
`args.local`	An optional list to be passed to user-defined local statistics that require additional arguments. By default `args.local = NULL`.
`args.global`	An optional list to be passed to global statistics that require additional arguments. For two-sided local statistics, `args.global` = list(one.sided=F) allows bi-directional differential expression to be considered.
`error`	Specifies the method for computing error rate estimates. "FDR.YB" computes the Yekutieli-Benjamini FDR estimate, "FWER.WY" computes the Westfall-Young FWER estimate. A Bonferroni, ("FWER.Bonf"), Holm's step-up ("FWER.Holm"), and Benjamini-Hochberg step down ("FDR.BH") adjustment can also be specified. By default ("none") no error rates are computed.
`alpha`	Allows the user to define the criterion for significance. By default, alpha will be 0.05 for nominal p-values (`error` = "none"), and 0.1 otherwise.
`method`	Type of hypothesis test can be specified as "permutation", "bootstrap.t", and "bootstrap.q". See vignette for details
`min.size`	Optional minimum category size to be considered.
`max.size`	Optional maximum category size to be considered.
`...`	Allows arguments from version 1.0 to be ignored

Details

`safe` utilizes a general framework for testing differential expression across gene categories that allows it to be used in various experimental designs. Through structured resampling of the data, `safe` accounts for the unknown correlation among genes, and enables proper estimation of error rates when testing multiple categories. `safe` also provides statistics and empirical p-values for the gene-specific differential expression.

Value

The function returns an object of class `SAFE`. See help for `SAFE-class` for more details.

Author(s)

William T. Barry: `<bill.barry@duke.edu>`

References

W. T. Barry, A. B. Nobel and F.A. Wright, 2005, *Significance Analysis of functional categories in gene expression studies: a structured permutation approach*, *Bioinformatics* **21**(9) 1943–1949.

See also the vignette included with this package.

See Also

`safeplot`, `getCmatrix`, `getPImatrix`.

Examples

```
## Simulate a dataset with 1000 genes and 20 arrays in a 2-sample design.
## The top 100 genes will be differentially expressed at varying levels

g.alt <- 100
g.null <- 900
n <- 20

data<-matrix(rnorm(n*(g.alt+g.null)),g.alt+g.null,n)
data[1:g.alt,1:(n/2)] <- data[1:g.alt,1:(n/2)] +
                         seq(2,2/g.alt,length=g.alt)
dimnames(data) <- list(c(paste("Alt",1:g.alt),
                         paste("Null",1:g.null)),
                       paste("Array",1:n))

## A treatment vector
trt <- rep(c("Trt","Ctr"),each=n/2)

## 2 alt. categories and 18 null categories of size 50

C.matrix <- kronecker(diag(20),rep(1,50))
dimnames(C.matrix) <- list(dimnames(data)[[1]],
    c(paste("TrueCat",1:2),paste("NullCat",1:18)))
dim(C.matrix)

results <- safe(data,trt,C.matrix,Pi.mat = 100)
results

## SAFE-plot made for the first category
if (interactive()) {
safeplot(results,"TrueCat 1")
}
```

`safedag` *SAFE results displayed in Gene Ontology*

Description

SAFE results are displayed on the directed acyclic graph for one of the ontologies under investigation. Category-wide significance displayed by node color.

Usage

```
safedag(object = NULL, ontology = NULL, top = NULL, file = NULL,
        color.cutoffs = c(0.1, 0.01, 0.001), filter = 0, max.GOnames = 200)
```

Arguments

`object`	Object of class `SAFE`
`ontology`	Gene Ontology of interest. Character strings of "GO.CC", "GO.BP", and "GO.MF" accepted.
`top`	Optional character string giving the node name from which to draw a subgraph of the tree
`file`	Optional filename for a post-script of the graph
`color.cutoffs`	Numeric vector of length 3 for the cutoffs for coloring significant nodes. Nodes with unadjusted p-values less than `color.cutoff[3]` are drawn in blue; less than `color.cutoff[2]` are drawn in green; less than `color.cutoff[1]` are drawn in red.
`filter`	Optional integer (1,2,3) to only include branches that contain at least one node as significant as the respective color.cutoff.
`max.GOnames`	Maximum size of DAG to include category names as labels.

Details

DAG-plots are suggested as a means for visualizing the extent of differential expression in Gene Ontology categories. The relatedness of significant categories suggests whether similar or disparate biological findings are identified.

Author(s)

William T. Barry: `<bill.barry@duke.edu>`

References

W. T. Barry, A. B. Nobel and F.A. Wright, 2005, *Significance Analysis of functional categories in gene expression studies: a structured permutation approach*, *Bioinformatics* **21**(9) 1943–1949.

See also the vignette included with this package.

See Also

`safe`.

`safeplot` *SAFE plot*

Description

A SAFE plot for a given category displays the empirical distribution function for the ranked local statistics of a given category.

Usage

```
safeplot(safe = NULL, cat.name = NULL, c.vec = NULL, local.stats = NULL,
         p.val = NULL, one.sided = NA, limits = c(-Inf,Inf), extreme = NA,
         italic = FALSE, x.label = "Ranked local statistic")
```

Usage

```
safeplot(safe)
safeplot(safe , cat.name)
safeplot(c.vec=, local.stats= , p.val=, one.sided=, limits=,
         extreme=, italic =, x.label=)
```

Arguments

`safe`	Object of class `SAFE`.
`cat.name`	Name of the category to be plotted. If omitted, the most significant category is plotted.
`c.vec`	Optional logical vector specifying membership to a gene category.
`local.stats`	Optional numeric vector of local statistics. Gene names should be provided as `names(local.stats)`.
`p.val`	Optional numeric value of the category's empirical p-value
`one.sided`	Optional logical value indicating if local statistics are one-sided.
`limits`	Limits of the shaded region in the plot on the unranked scale.
`extreme`	Optional logical value whether only genes in the shaded region should be labeled.
`italic`	Optional logical value whether gene names should be italic.
`x.label`	Character string for the x-axis label.

Details

SAFE-plots are suggested as appropriate for visualizing the differential expression in a given category relative to the complementary set of genes. The empirical cumulative distribution is plotted for the ranked local statistics in the category. Tick marks are drawn along the top of the graph to indicate each gene's positions, and labeled when sufficient space permits. In this manner, genes with the most extreme local statistics can be identified as contributing to a categories significance.

Author(s)

William T. Barry: `<bill.barry@duke.edu>`

References

W. T. Barry, A. B. Nobel and F.A. Wright, 2005, *Significance Analysis of functional categories in gene expression studies: a structured permutation approach*, *Bioinformatics* **21**(9) 1943–1949.

See also the vignette included with this package.

See Also

`safe`.

Examples

```
## Simulate a dataset with 1000 genes and 20 arrays in a 2-sample design.
## The top 100 genes will be differentially expressed at varying levels

g.alt <- 100
g.null <- 900
n <- 20

data<-matrix(rnorm(n*(g.alt+g.null)),g.alt+g.null,n)
data[1:g.alt,1:(n/2)] <- data[1:g.alt,1:(n/2)] +
                         seq(2,2/g.alt,length=g.alt)
dimnames(data) <- list(c(paste("Alt",1:g.alt),
                         paste("Null",1:g.null)),
                       paste("Array",1:n))

## A treatment vector
trt <- rep(c("Trt","Ctr"),each=n/2)

## 2 alt. categories and 18 null categories of size 50

C.matrix <- kronecker(diag(20),rep(1,50))
dimnames(C.matrix) <- list(dimnames(data)[[1]],
    c(paste("TrueCat",1:2),paste("NullCat",1:18)))
dim(C.matrix)

results <- safe(data,trt,C.matrix,Pi.mat = 100)
results

## SAFE-plot made for the first category
if (interactive()) {
safeplot(results,"TrueCat 1")
}
```

Index