Package 'GOexpress'

April 10, 2015

Title Visualise microarray and RNAseq data using gene ontology annotations

Version 1.0.1

Date 2014-10-17

Description The package contains methods to visualise the expression levels of genes from a microarray or RNA-seq experiment and offers a clustering analysis to identify GO terms enriched in genes with expression levels best clustering two or more predefined groups of samples. Annotations for the genes present in the expression dataset are obtained from Ensembl through the biomaRt package. The random forest framework is used to evaluate the ability of each gene to cluster samples according to the factor of interest. Finally, GO terms are scored by averaging the rank (alternatively, score) of their respective gene sets to cluster the samples. An ANOVA approach is also available as an alternative statistical framework.

Depends R (>= 3.0.2), grid, Biobase (>= 2.22.0)

Imports biomaRt (>= 2.18.0), stringr (>= 0.6.2), ggplot2 (>= 0.9.0), RColorBrewer (>= 1.0), gplots (>= 2.13.0), VennDiagram (>= 1.6.5), randomForest (>= 4.6)

License GPL (>= 3)

biocViews Software, GeneExpression, Transcription, DifferentialExpression, GeneSetEnrichment, DataRepresentation, Metagenomics, Clustering, TimeCourse, Microarray, Sequencing, RNASeq, Annotation, MultipleComparison, Pathways, GO, Visualization

URL `https://github.com/kevinrue/GOexpress-release`

LazyData true

Author Kevin Rue-Albrecht [aut, cre],
Paul A. McGettigan [ctb],
Belinda Hernandez [ctb],
David A. Magee [ctb],
Nicolas C. Nalpas [ctb],
Andrew Parnell [ctb],

Stephen V. Gordon [ths],
David E. MacHugh [ths]

Maintainer Kevin Rue-Albrecht <kevin.rue@ucdconnect.ie>

R topics documented:

`GOexpress-package` *Visualise microarray and RNAseq data with gene ontology annotations.*

Description

Combines local gene expression data with online Gene Ontology (GO) resources to visualise and rank ontologies enriched for genes best clustering predefined groups of samples based on gene expression levels. Using the `biomaRt` package, `GOexpress` semi-automatically retrieves gene ontology annotations from Ensembl for the species corresponding to the expression dataset prior to performing the analysis. A random forest analysis is performed to evaluate the ability of each gene to cluster samples according to a predefined grouping factor (one-way ANOVA available as an alteranative). Each GO term is scored and ranked according to the average rank (alternatively, average power) of all associated genes to cluster the samples according to the factor. The ranked list of GO terms is returned, with tools allowing to visualise the statistics on a gene- and ontology-basis.

Details

Package:	GOexpress
Type:	Package
Version:	1.0.1
Date:	2014-10-17
License:	GPL (>= 3)

This package requires only two input variables

1. An ExpressionSet containing assayData and phenoData. The former should be a gene-by-sample matrix providing gene expression values for each gene in each sample. The latter should be an AnnotatedDataFrame from the Biobase package providing phenotypic information and grouping factors with two or more levels.
2. The name of the grouping factor to investigate, which must be a valid column name in the phenoData.

Following analysis, visualisation methods include:

- Histogram and quantiles representations of the scores of GO terms
- Filtering of results on various criteria (e.g. number of genes annotated to GO term)
- Re-ordering of GO terms and gene result tables based on score or rank metric
- Table of statistics for genes annotated to a given GO term
- Hierarchical clustering of samples based on the expression level of genes annotated to a given GO term
- Heatmap of samples and genes based on the expression level of genes annotated to a given GO term
- Expression profile of a gene against one given factor (e.g. Time) while grouping samples on another given factor (e.g. Treatment)
- Univariate analysis of the expression level of a gene in the different groups of each experimental factor.
- Venn diagram of the counts of genes shared between a list of GO terms.

Author(s)

Kevin Rue-Albrecht

Maintainer: Kevin Rue-Albrecht <kevin.rue@ucdconnect.ie>

See Also

Main method for an example usage: `GO_analyse`.

Packages `Biobase`, `ggplot2`, `randomForest`, `RColorBrewer`, `VennDiagram`.

Methods `getBM`, `heatmap.2`, `bluered`, `greenred`, `grid.newpage`, `grid.layout`, `str_extract`.

Examples

```
# Sample input data available with package:
data(AlvMac)

# Sample output data available with package:
data(AlvMac_results)

# Supported species and microarrays:
data(microarray2dataset)
data(prefix2dataset)
```

AlvMac — *Sample data from a RNAseq experiment.*

Description

An example ExpressionSet including expression data and phenotypic information about the samples.

The expression data is saved in the assayData slot of the ExpressionSet. It is a gene-by-sample matrix, containing a subset of data from an *in vitro* stimulation of bovine macrophages with different mycobacterial strains. Column names are sample names, and row names are Ensembl gene identifiers of the *Bos taurus* species. Each cell contains the log2-transformed normalised expression level of each gene in each sample.

The phenotypic information is saved in the phenoData slot of the ExpressionSet. Row names are sample names and columns contain descriptive information about each sample, including experimental factors(e.g. Treatment, Timepoint, Animal).

Usage

```
data(AlvMac)
```

Details

Gene expression was measured in poly-A purified strand-specific RNA libraries using the RNA-Sequencing Illumina(R) HiSeq(R) 2000 platform as paired-end 2 x 90 nucleotide reads. Raw reads from pooled RNA libraries were first deconvoluted according to sample-specific nucleotide barcodes. Read pairs containing adapter sequence in either read mate were discarded, and similarly read pairs of low overall quality in either mate were also discarded. Paired-end reads from each filtered individual library were aligned to the *Bos taurus* reference genome (*B. taurus* UMD3.1.71 genome release) using the STAR aligner software. For each library, raw counts for each gene based on sense strand data were obtained using the featureCounts software from the Subread package. The featureCounts parameters were set to unambiguously assign uniquely aligned paired-end reads in a stranded manner to the exons of genes within the *Bos taurus* reference genome annotation (*B. taurus* UMD3.1.71 genome annotation). The gene count outputs were further processed using the edgeR Bioconductor package.

The gene expression quantitation pipeline within the edgeR package was customised to: (1) filter out all bovine rRNA genes; (2) filter out genes displaying expression levels below the minimally-set threshold of one count per million [CPM] in at least ten individual libraries (number of biological replicates); (3) calculate normalisation factors for each library using the trimmed mean of M-values method; (4) log2-transform CPM values based on the normalised library size.

To generate this test data subset, we extracted 100 genes from the original dataset of 12,121 genes. All 7 genes associated with the GO term "GO:0034142" (i.e. "toll-like receptor 4 signaling pathway") were kept, plus another random 93 random genes, making a total of 100 genes measured in 117 samples. Samples include all 10 biological replicates collected at four different time-points, see `data(targets)`. The TLR4 pathway was found in the full dataset as the top-ranking biological pathway discriminating the different mycobacterial infections (unpublished observations).

Value

`assayData` is a matrix of expression levels for 100 genes (rows) measured in 117 samples (columns).

- `rownames` are Ensembl gene identifiers of the *Bos taurus* species.
- `colnames` are samples identifiers.

`phenoData` is a data frame with 117 samples and 7 descriptive fields (e.g. experimental factors) in the columns listed below:

- `rownames` are unique identifiers. Here, sample names.
- `File` contains local filenames where the RNAseq counts were obtained from.
- `Sample` contains individual sample name.
- `Animal` contains the unique identifier of the animal corresponding to the biological replicate, stored as a factor.
- `Treatment` contains the infection status of the sample, stored as a factor (CN: Control, MB: *M. bovis*, TB: *M. tuberculosis*)
- `Time` contains the time of measurement in hours post-infection,stored as a factor.
- `Group` contains a combination of the Treatment and Time factors above, stored as a factor itself.
- `Timepoint` contains the time of measurement, stored as a numeric value. This field is useful to use on the X-axis of expression plots. See function `expression_plot()`.

Source

Publication in review process.

Examples

```
# Load the data
data(AlvMac)

# Structure of the data
str(AlvMac)

# Dimensions (rows, columns) of the data
dim(AlvMac)
```

```
# Subset of first 5 features and 5 samples
AlvMac[1:5, 1:5]

# Phenotypic information
pData(AlvMac)

# Phenotypic information about factor "Group"
pData(AlvMac)$Group

# Conversion of a factor to a character vector
as.character(pData(AlvMac)$Group)

# Number of samples (rows) and annotations (columns)
dim(pData(AlvMac))
```

`AlvMac_results` *Sample output from the* `GO_analyse()` *function on an RNAseq experiment.*

Description

This variable may be used to test the filtering and visualisation methods implemented in the package. It contains the output of the command `AlvMac_results = GO_analyse(eSet=AlvMac, f="Treatment")` applied to the sample input data `data(AlvMac)`.

Usage

```
data(AlvMac_results)
```

Value

A list of 7 slots summarising the input and results of the analysis:

- `GO` contains a table ranking all GO terms related to genes in the expression dataset based on the average ability of their related genes to cluster the samples according to the predefined grouping factor.
- `mapping` contains the table mapping genes present in the dataset to GO terms.
- `genes` contains a table ranking all genes present in the expression dataset based on their ability to cluster the samples according to the predefined grouping factor (see 'factor' below).
- `factor` contains the grouping factor analysed.
- `method` contains the statistical framework used.
- `ntree` contains number of trees built during the randomForest analysis.
- `mtry` contains the number of features randomly sampled as candidates at each split in each tree built during the randomForest analysis.

Warning

Running the above command again, you might obtain slightly different scores and ranks due to the stochastic process of sampling used by the random forest algorithm. However, the ranking metric was found to be robust and stable across run, given adequate number of trees and predictor variables sampled.

Source

Source data are part of a publication in review.

Examples

```
data(AlvMac_results)
str(AlvMac_results)
head(AlvMac_results$GO, n=20)
head(AlvMac_results$GO$genes, n=20)
```

`cluster_GO` *Generates a hierarchical clustering of the samples*

Description

Clusters the samples using only the expression levels of genes associated with a given go_id.

Usage

```
cluster_GO(
    go_id, result, eSet, f=result$factor, subset=NULL,
    method_dist="euclidean", method_hclust="average", cex=0.8,
    main=paste(go_id, result$GO[result$GO$go_id == go_id, "name_1006"]),
    xlab="Distance", cex.main=1, main.Lsplit=NULL, ...)
```

Arguments

`go_id`	A Gene Ontology (GO) identifier.
`result`	The output of `GO_analyse()` or a subset of it obtained from `subset_scores()`.
`eSet`	`ExpressionSet` of the `Biobase` package including a gene-by-sample expression matrix in the `AssayData` slot, and a phenotypic information data-frame in the `phenodata` slot. In the expression matrix, row names are Ensembl gene identifiers or probeset identifiers, and column names are sample identifiers. In the phentypic data-frame, row names are sample idenfifiers, column names are grouping factors and phenotypic traits usable for the one-way ANOVA.
`f`	The grouping factor in `phenodata` to label the samples by.
`subset`	A named list to subset `eSet`. Names must be column names existing in colnames(pData(eSet)). Values must be vectors of values existing in the corresponding column of pData(eSet).

`method_dist`	The method used to calculate distance between samples. See the `dist()` method from package `stats`.
`method_hclust`	The method used to cluster samples. See the `hclust()` method from package `stats`.
`cex`	A numeric value defining the character expansion of text in the plot.
`main`	A character string for the main title of the plot.
`xlab`	A label for the x axis, defaults to "Distance".
`cex.main`	Scaling factor of the main title font size. Default is 1. We suggest to use it in combination with the argument `main.Lsplit` for GO terms with long names.
`main.Lsplit`	Number of characters after which a new-line character will be inserted in the main title. If this would occur within a word, the new-line character will be inserted before this word. Default is NULL, leaving the title on a single line.
`...`	Additional parameters passed on to `dist()`, `hclust()` and `plot()`.

Value

Returns the output of the `plot()` function.

Author(s)

Kevin Rue-Albrecht

See Also

Method `GO_analyse`.

Examples

```
# load the sample output data
data(AlvMac_results)

# Hierarchical clustering of samples based on the same GO term
cluster_GO(go_id="GO:0034142", result=AlvMac_results, eSet=AlvMac, cex=0.7)

# Re-label sample by another factor
cluster_GO(go_id="GO:0034142", result=AlvMac_results, eSet=AlvMac, cex=0.7,
           f="Group")
```

expression_plot — *Plots the expression profile of a gene by levels of a factor*

Description

This function will plot the expression profile of a gene across a valid X-axis variable in the phenodata while representing the mean and confidence interval of groups of samples defined by levels of another valid grouping factor in the phenodata.

Usage

```
expression_plot(
    gene_id, result, eSet, x_var, f=result$factor, subset=NULL,
    xlab=x_var, ylab="log2(cpm)", ylim=range(exprs(eSet)),
    col.palette="Accent",
    col=brewer.pal(n=length(levels(pData(eSet)[,f])), name=col.palette),
    level=0.95, title=NULL, title.size=2, axis.title.size=20,
    axis.text.size=15, axis.text.angle=0,
    legend.title.size=20, legend.text.size=15, legend.key.size=30)
```

Arguments

gene_id	An gene or probeset identifier present in rownames(expr_data).
result	An output of the GO_analyse() or subset_scores() function.
eSet	ExpressionSet of the Biobase package including a gene-by-sample expression matrix in the AssayData slot, and a phenotypic information data-frame in the phenodate slot. In the expression matrix, row names are Ensembl gene identifiers or probeset identifiers, and column names are sample identifiers. In the phentypic data-frame, row names are sample idenfifiers, column names are grouping factors and phenotypic traits usable for the one-way ANOVA.
x_var	A column name in phenodata to plot on the X-axis. If representing time on the X-axis, users should store the time-points as numeric values in the AnnotatedDataFrame to adequately space the time-points.
f	A column name in phenodata to group the samples when representing mean and confidence interval. The factor specified in the initial GO_analyse() call is used by default. Unexpected grouping factors of samples can reveal interesting trends (e.g. "Animal", "Tissue", "CellType", ...).
subset	A named list to subset eSet. Names must be column names existing in colnames(pData(eSet)). Values must be vectors of values existing in the corresponding column of pData(eSet).
xlab	Title of the X-axis. Default is tha value of x_var.
ylab	Title of the Y-axis. Default is "log2(cpm)".
ylim	Numeric vector of length 2 specifying the lower and upper bounds of the Y axis. Default is scaled to the full range of expression values in the expression dataset, to ease comparison of different genes. If set to NULL, the axis will be scaled to fit the plotted data only.

`col.palette`	A valid `RColorBrewer` palette name to fetch the colormap from. Default is palette "Accent".
`col`	A vector of color names or codes. The number of colors provided must match the number of levels of the grouping factor. If specified, overrides argument `col.palette`.
`level`	The confidence interval level to visualise around the mean of each group. Default is 0.95.
`title`	Changes the plot title. Default is a combination of the gene id and the associated gene.
`title.size`	Changes the font size of the title. Default is 2.
`axis.title.size`	Changes the font size of the axes title. Defalt is 20.
`axis.text.size`	Changes the font size of the axes text labels. Default is 15.
`axis.text.angle`	Changes the angle of the X axis text labels. Default is 0 (horizontal).
`legend.title.size`	Changes the font size of the legend title. Default is 20.
`legend.text.size`	Changes the font size of the legend text labels. Default is 15.
`legend.key.size`	Changes the size of the legend keys (in points). Default is 30.

Value

The ggplot object.

Warning

Common issues:

- It may not be possible to produce plots where the combination of X-axis variable and grouping factor leaves too few replicates to compute a confidence interval for each X value. This is a limitation imposed by the ggplot2 package to produce proper statistics and confidence intervals. In such cases, it may be preferrable to use the expression_profiles() method.

Author(s)

Kevin Rue-Albrecht

References

- ggplot2 package.

See Also

Packages `Biobase` and `ggplot2` , methods `expression_plot_symbol` and `GO_analyse`.

Examples

```
# load the sample output data
data(AlvMac_results)

# Expression by gene identifier (TNIP3)
expression_plot(gene_id="ENSBTAG00000047107", eSet=AlvMac, x_var="Timepoint",
                result=AlvMac_results)

# Same gene, plotted by animal and grouped by treatment (merging time points)
expression_plot(gene_id="ENSBTAG00000047107", eSet=AlvMac, x_var="Animal",
                result=AlvMac_results,
                f="Treatment")

# Same gene, plotted by animal and grouped by time-point (merging treatments)
expression_plot(gene_id="ENSBTAG00000047107", eSet=AlvMac, x_var="Animal",
                result=AlvMac_results,
                f="Time")
```

---

expression_plot_symbol

Plots the expression profile of a gene by levels of a factor

Description

This function will plot the expression profile of a gene across a valid X-axis variable from the `AnnotatedDataFrame` while representing the mean and confidence interval of groups of samples defined by levels of a valid grouping factor from the AnnotatedDataFrame.

In the case of a gene name represented by multiple gene or probeset identifiers in the dataset, a lattice of plots will be produced. Each of the plots in this lattice can subsequently be plotted separately using its associated index.

Usage

```
expression_plot_symbol(
    gene_symbol, result, eSet, x_var, f=result$factor, subset=NULL,
    index=0, xlab=x_var, ylab="log2(cpm)", ylim=range(exprs(eSet)),
    col.palette="Accent",
    col=brewer.pal(n=length(levels(pData(eSet)[,f])), name=col.palette),
    level=0.95, titles=c(), title.size=2, axis.title.size=20,
    axis.text.size=15, axis.text.angle=0,
    legend.title.size=20, legend.text.size=20, legend.key.size=30)
```

Arguments

`gene_symbol`	A gene name present in `rownames(AlvMac_results$genes)`.
`result`	An output of the `GO_analyse()` or `subset_scores()` function.

`eSet`	`ExpressionSet` of the `Biobase` package including a gene-by-sample expression matrix in the `AssayData` slot, and a phenotypic information data-frame in the `phenodate` slot. In the expression matrix, row names are Ensembl gene identifiers or probeset identifiers, and column names are sample identifiers. In the phentypic data-frame, row names are sample idenfifiers, column names are grouping factors and phenotypic traits usable for the one-way ANOVA.
`x_var`	A column name in `phenodata` to plot on the X-axis. If representing time on the X-axis, users should store the time-points as numeric values in the `AnnotatedDataFrame` to adequately space the time-points.
`f`	A column name in `phenodata` to group the samples when representing mean and confidence interval. The factor specified in the initial `GO_analyse()` call is used by default. Unexpected grouping factors of samples can reveal interesting trends (e.g. "Animal", "Tissue", "CellType", ...).
`subset`	A named list to subset `eSet`. Names must be column names existing in colnames(pData(eSet)). Values must be vectors of values existing in the corresponding column of pData(eSet).
`index`	In the case where multiple gene or probeset identifiers are associated with the gene name, `index=2` will plot the expression profile of the second feature identifier alone, for instance.
`xlab`	Title of the X-axis. Default is tha value of `x_var`.
`ylab`	Title of the Y-axis. Default is "log2(cpm)".
`ylim`	Numeric vector of length 2 specifying the lower and upper bounds of the Y axis. Default is scaled to the full range of expression values in the expression dataset, to ease comparison of different genes. If set to NULL, the axis will be scaled to fit the plotted data only.
`col.palette`	A valid `RColorBrewer` palette name to fetch the colormap from.
`col`	A vector of color names or codes. The number of colors provided must match the number of levels of the grouping factor. Default to a palette with an adequate set of colors. If specified, overrides argument `col.palette`.
`level`	The confidence interval level to visualise around the mean of each group. Default is 0.95.
`titles`	Character vector providing as many titles as there are plots to replace the default titles. Default is a combination of the gene id and the associated gene.
`title.size`	Changes the font size of the title. Default is 2.
`axis.title.size`	Changes the font size of the axes title. Defalt is 20.
`axis.text.size`	Changes the font size of the axes text labels. Default is 15.
`axis.text.angle`	Changes the angle of the X axis text labels. Default is 0 (horizontal).
`legend.title.size`	Changes the font size of the legend title. Default is 20.
`legend.text.size`	Changes the font size of the legend text labels. Default is 15.
`legend.key.size`	Changes the size of the legend keys (in points). Default is 30.

Value

The ggplot object, or NULL if multiple features are annotated to the gene symbol.

Warning

Common issues:

- It may not be possible to produce plots where the combination of X-axis variable and grouping factor leaves too few replicates to compute a confidence interval for each X value. This is a limitation imposed by the ggplot2 package to produce proper statistics and confidence intervals. In such cases, it may be preferrable to use the expression_profiles_symbol() method.

Author(s)

Kevin Rue-Albrecht

References

- ggplot2 package.

See Also

Packages `Biobase` and `ggplot2` , methods `expression_plot` and `GO_analyse`.

Examples

```
# load the sample output data
data(AlvMac_results)

# Expression by gene identifier (TNIP3)
expression_plot_symbol(gene_symbol="PIK3AP1", eSet=AlvMac, x_var="Timepoint",
                        result=AlvMac_results)

# Same gene, plotted by animal and grouped by treatment (merging time points)
expression_plot_symbol(gene_symbol="PIK3AP1", eSet=AlvMac, x_var="Animal",
                        result=AlvMac_results,
                        f="Treatment")

# Same gene, plotted by animal and grouped by time-point (merging treatments)
expression_plot_symbol(gene_symbol="PIK3AP1", eSet=AlvMac, x_var="Animal",
                        result=AlvMac_results,
                        f="Time")
```

expression_profiles *Plots the individual expression profile of a gene in samples series*

Description

This function will plot the expression profile of a gene in individual samples series across a valid X-axis variable in the phenodata, while colouring sample groups according to another variable in the phenodata, and using different line types according to yet another (or the same) variable in the phenodata.

Usage

```
expression_profiles(
    gene_id, result, eSet, x_var, seriesF, subset=NULL,
    colourF=result$factor, linetypeF=colourF, line.size=1.5,
    xlab=x_var, ylab="log2(cpm)", ylim=range(exprs(eSet)),
    col.palette="Accent",
    col=brewer.pal(n=length(levels(pData(eSet)[,colourF])),
                   name=col.palette),
    lty=1:length(levels(pData(eSet)[,linetypeF])),
    title=NULL, title.size=2, axis.title.size=20,
    axis.text.size=15, axis.text.angle=0,
    legend.title.size=20, legend.text.size=15,
    legend.key.size=30)
```

Arguments

`gene_id`	An gene or probeset identifier present in `rownames(expr_data)`.
`result`	An output of the `GO_analyse()` or `subset_scores()` function.
`eSet`	`ExpressionSet` of the `Biobase` package including a gene-by-sample expression matrix in the `AssayData` slot, and a phenotypic information data-frame in the `phenodate` slot. In the expression matrix, row names are Ensembl gene identifiers or probeset identifiers, and column names are sample identifiers. In the phentypic data-frame, row names are sample idenfifiers, column names are grouping factors and phenotypic traits usable for the one-way ANOVA.
`x_var`	A column name in `phenodata` to plot on the X-axis. If representing time on the X-axis, users should store the time-points as numeric values in the `AnnotatedDataFrame` to adequately space the time-points.
`seriesF`	A column name in `phenodata` to track samples of an identical series across the X-axis. The combination of `seriesF` and `x_var` should uniquely identify each sample in `eSet`.
`subset`	A named list to subset `eSet`. Names must be column names existing in colnames(pData(eSet)). Values must be vectors of values existing in the corresponding column of pData(eSet).
`colourF`	A column name in `phenodata` to colour the corresponding groups of samples. The factor specified in the initial `GO_analyse()` call is used by default.

`linetypeF`	A column name in `phenodata` to assign a different line type to the corresponding groups of samples. Default mirrors `colourF`.
`line.size`	The width of the plotted lines. Default is 1.5.
`xlab`	Title of the X-axis. Default is tha value of `x_var`.
`ylab`	Title of the Y-axis. Default is "log2(cpm)".
`ylim`	Numeric vector of length 2 specifying the lower and upper bounds of the Y axis. Default is scaled to the full range of expression values in the expression dataset, to ease comparison of different genes. If set to NULL, the axis will be scaled to fit the plotted data only.
`col.palette`	A valid `RColorBrewer` palette name to fetch the colormap from. Default is palette "Accent".
`col`	A vector of color names or codes. The number of colors provided must match the number of levels of the factor `colourF`. If specified, overrides argument `col.palette`. Default colours are obtained from the `col.palette`.
`lty`	A vector of numeric values corresponding to line types. The number of line types provided must match the number of levels of the factor `linetypeF`. Default line types are `1:length(levels(pData(eSet)[, linetypeF]))`.
`title`	Changes the plot title. Default is a combination of the gene id and the associated gene.
`title.size`	Changes the font size of the title. Default is 2.
`axis.title.size`	Changes the font size of the axes title. Defalt is 20.
`axis.text.size`	Changes the font size of the axes text labels. Default is 15.
`axis.text.angle`	Changes the angle of the X axis text labels. Default is 0 (horizontal).
`legend.title.size`	Changes the font size of the legend title. Default is 20.
`legend.text.size`	Changes the font size of the legend text labels. Default is 15.
`legend.key.size`	Changes the size of the legend keys (in points). Default is 30.

Details

In order to track and visualise individual sample series, each sample in the `ExpressionSet` should be associated with a unique combination of `seriesF` and `x_var`. This may require the generation of a new factor in the `phenodata`, combining all experimental factors except that plotted on the X-axis. See below for an example on the training dataset.

Value

The ggplot object.

Author(s)

Kevin Rue-Albrecht

References

- ggplot2 package.

See Also

Packages Biobase and ggplot2 , methods expression_plot_symbol and expression_plot_symbol.

Examples

```
# load the sample output data
data(AlvMac_results)

AlvMac$Series = paste(AlvMac$Animal, AlvMac$Treatment, sep="_")

expression_profiles(gene_id = "ENSBTAG00000047107", result = AlvMac_results,
                    eSet=AlvMac, x_var = "Timepoint", seriesF="Series",
                    legend.title.size=10, legend.text.size=10,
                    legend.key.size=15)

expression_profiles(gene_id = "ENSBTAG00000047107", result = AlvMac_results,
                    eSet=AlvMac, x_var = "Timepoint", seriesF="Series",
                    linetypeF="Animal",
                    legend.title.size=10, legend.text.size=10,
                    legend.key.size=15)
```

expression_profiles_symbol

Plots the individual expression profile of a gene in samples series

Description

This function will plot the expression profile of a gene in individual samples series across a valid X-axis variable in the phenodata, while colouring sample groups according to another variable in the phenodata, and using different line types according to yet another (or the same) variable in the phenodata.

Usage

```
expression_profiles_symbol(
    gene_symbol, result, eSet, x_var, seriesF, subset=NULL,
    colourF=result$factor, linetypeF=colourF, line.size=1.5,
    index=0, xlab=x_var, ylab="log2(cpm)", ylim=range(exprs(eSet)),
    col.palette="Accent",
    col=brewer.pal(n=length(levels(pData(eSet)[,colourF])),
                   name=col.palette),
    lty=1:length(levels(pData(eSet)[,linetypeF])),
    titles=c(), title.size=2, axis.title.size=20,
    axis.text.size=15, axis.text.angle=0,
    legend.title.size=20, legend.text.size=15,
    legend.key.size=30)
```

Arguments

`gene_symbol`	A gene name present in `rownames(AlvMac_results$genes)`.
`result`	An output of the `GO_analyse()` or `subset_scores()` function.
`eSet`	`ExpressionSet` of the `Biobase` package including a gene-by-sample expression matrix in the `AssayData` slot, and a phenotypic information data-frame in the `phenodate` slot. In the expression matrix, row names are Ensembl gene identifiers or probeset identifiers, and column names are sample identifiers. In the phentypic data-frame, row names are sample idenfifiers, column names are grouping factors and phenotypic traits usable for the one-way ANOVA.
`x_var`	A column name in `phenodata` to plot on the X-axis. If representing time on the X-axis, users should store the time-points as numeric values in the `AnnotatedDataFrame` to adequately space the time-points.
`seriesF`	A column name in `phenodata` to track samples of an identical series across the X-axis. The combination of `seriesF` and `x_var` should uniquely identify each sample in `eSet`.
`subset`	A named list to subset `eSet`. Names must be column names existing in colnames(pData(eSet)). Values must be vectors of values existing in the corresponding column of pData(eSet).
`colourF`	A column name in `phenodata` to colour the corresponding groups of samples. The factor specified in the initial `GO_analyse()` call is used by default.
`linetypeF`	A column name in `phenodata` to assign a different line type to the corresponding groups of samples. Default mirrors `colourF`.
`line.size`	The width of the plotted lines. Default is 1.5.
`index`	In the case where multiple gene or probeset identifiers are associated with the gene name, `index=2` will plot the expression profile of the second feature identifier alone, for instance.
`xlab`	Title of the X-axis. Default is tha value of `x_var`.
`ylab`	Title of the Y-axis. Default is "log2(cpm)".
`ylim`	Numeric vector of length 2 specifying the lower and upper bounds of the Y axis. Default is scaled to the full range of expression values in the expression dataset, to ease comparison of different genes. If set to NULL, the axis will be scaled to fit the plotted data only.
`col.palette`	A valid `RColorBrewer` palette name to fetch the colormap from. Default is palette "Accent".
`col`	A vector of color names or codes. The number of colors provided must match the number of levels of the factor `colourF`. If specified, overrides argument `col.palette`. Default colours are obtained from the `col.palette`.
`lty`	A vector of numeric values corresponding to line types. The number of line types provided must match the number of levels of the factor `linetypeF`. Default line types are `1:length(levels(pData(eSet)[, linetypeF]))`.

`titles`	Character vector providing as many titles as there are plots to replace the default titles. Default is a combination of the gene id and the associated gene.
`title.size`	Changes the font size of the title. Default is 2.
`axis.title.size`	Changes the font size of the axes title. Defalt is 20.
`axis.text.size`	Changes the font size of the axes text labels. Default is 15.
`axis.text.angle`	Changes the angle of the X axis text labels. Default is 0 (horizontal).
`legend.title.size`	Changes the font size of the legend title. Default is 20.
`legend.text.size`	Changes the font size of the legend text labels. Default is 15.
`legend.key.size`	Changes the size of the legend keys (in points). Default is 30.

Details

In order to track and visualise individual sample series, each sample in the `ExpressionSet` should be associated with a unique combination of `seriesF` and `x_var`. This may require the generation of a new factor in the `phenodata`, combining all experimental factors except that plotted on the X-axis. See below for an example on the training dataset.

Value

The ggplot object, or NULL if multiple features are annotated to the gene symbol.

Author(s)

Kevin Rue-Albrecht

References

- ggplot2 package.

See Also

Packages `Biobase` and `ggplot2` , methods `expression_plot_symbol` and `expression_plot_symbol`.

Examples

```
# load the sample output data
data(AlvMac_results)

AlvMac$Series = paste(AlvMac$Animal, AlvMac$Treatment, sep="_")

expression_profiles_symbol(gene_symbol="TNIP3", result = AlvMac_results,
        linetypeF="Animal", line.size=1.5,
        eSet=AlvMac, seriesF="Series", x_var = "Timepoint",
        title.size=1.5, legend.title.size=10, legend.text.size=10,
        legend.key.size=30)

expression_profiles_symbol(gene_symbol="TNIP3", result = AlvMac_results,
        linetypeF="Animal", line.size=1.5, lty=rep(1,10),
        eSet=AlvMac, seriesF="Series", x_var = "Timepoint",
        title.size=1.5, legend.title.size=10, legend.text.size=10,
        legend.key.size=30)
```

GO_analyse — *Identifies gene ontologies clustering samples according to predefined factor.*

Description

Combines local gene expression data with online Gene Ontology (GO) resources to visualise and rank ontologies enriched for genes best clustering predefined groups of samples based on gene expression levels.

Using the `biomaRt` package, this method semi-automatically retrieves gene ontology annotations from Ensembl for the species corresponding to the expression dataset prior to performing the analysis. A random forest analysis is performed to evaluate the ability of each gene to cluster samples according to a predefined grouping factor (one-way ANOVA available as an alteranative). Each GO term is scored and ranked according to the average rank (alternatively, average power) of all associated genes to cluster the samples according to the factor. The ranked list of GO terms is returned, with tools allowing to visualise the statistics on a gene- and ontology-basis.

Usage

```
GO_analyse(
    eSet, f, subset=NULL, biomart_dataset="", microarray="",
    method="randomForest", rank.by="rank", do.trace=100, ntree=1000,
    mtry=ceiling(2*sqrt(nrow(eSet))),
    ...)
```

Arguments

`eSet`	`ExpressionSet` of the `Biobase` package including a gene-by-sample expression matrix in the `AssayData` slot, and a phenotypic information data-frame in the `phenodate` slot. In the expression matrix, row names are Ensembl gene identifiers or probeset identifiers, and column names are sample identifiers. In the phentypic data-frame, row names are sample idenfifiers, column names are grouping factors and phenotypic traits usable for the one-way ANOVA.
`f`	A column name in `phenodata` used as the grouping factor for the analysis.
`subset`	A named list to subset `eSet` for the analysis. Names must be column names existing in colnames(pData(eSet)). Values must be vectors of values existing in the corresponding column of pData(eSet). The original ExpressionSet will be left unchanged.

`biomart_dataset`	The BioMart Ensembl dataset corresponding to the species studied. Automatically detected if not specified. Use data(prefix2dataset) for valid values.
`microarray`	The identifier in the Ensembl BioMart corresponding to the microarray platform used. If not specified and Ensembl gene identifiers are not detected in the expression dataset, the method will attempt to automatically identify the platform used from the first probeset in the dataset. See `data(microarray2dataset)` for valid values.
`method`	Either "randomForest" or "rf" to use the random forest framework, or alternatively either of "anova" or "a" to use the one-way ANOVA model. Default is "randomForest".
`rank.by`	Either of "rank" or "score" to chose the metric used to order the gene and GO term result tables.
`do.trace`	Only used if method="randomForest". If set to TRUE, give a more verbose output as randomForest is run. If set to some integer, then running output is printed for every do.trace trees. Default is 100.
`ntree`	Only used if method="randomForest". Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times
`mtry`	Only used if method="randomForest". Number of features randomly sampled as candidates at each split. Default value is 2*sqrt(gene_count) which is approximately 220 genes for a dataset of 12,000 genes.
`...`	Additional arguments passed on to the randomForest() method.

Details

The current scoring functions strongly favor GO terms associated with fewer genes at the top of the ranking. This bias may actually be seen as a valuable feature which enables the user to browse through GO terms of increasing "granularity", i.e. associated with increasingly large sets of genes, although consequently being increasingly vague and blurry (e.g. "protein binding" molecular function associated with over 6,000 genes).

It is suggested to use the `subset_scores()` function to filter out GO terms with fewer than 5+ genes associated with it. Indeed, those GO terms are more sensitive to outlier genes as they were scored on the average of a handful of genes.

Value

A list containing the results of the analysis. Some elements are specific to each analysis method.

Core elements:

`GO`	A table ranking all GO terms related to genes in the expression dataset based on the average ability of their related genes to cluster the samples according to the predefined grouping factor.
`mapping`	The table mapping genes present in the dataset to GO terms.
`genes`	A table ranking all genes present in the expression dataset based on their ability to cluster the samples according to the predefined grouping factor.

`factor`	The predefined grouping factor.
`method`	The statistical framework used.
`subset`	The filters used to run the analysis only on a subet of the samples. NULL if no filter was applied.

Random Forest additional elements:

`ntree`	Number of trees grown.
`mtry`	Number of variables randomly sampled as candidates at each split.

One-way ANOVA does not have additional arguments.

Warning

Make sure that the factor `f` is an actual factor in the R language meaning. This is important for the underlying statistical framework to identify the groups of samples defined by their level of this factor.

If the column defining the factor (e.g. "Treatment") in codephenodata is not an R factor, use `pData(targets)$Treatment = factor(pData(targets)$Treatment)` to convert the character values into an actual R factor with appropriate levels.

Author(s)

Kevin Rue-Albrecht

See Also

Methods `subset_scores`, `getBM`, `randomForest`, and `oneway.test`.

Examples

```
# Load example data subset
data(AlvMac)

# Run the analysis on factor "Treatment",
# considering only treatments "MB" and "TB" at time-point "48H"
# Factor f must be one of the column names in pData(targets)
AlvMac_results <- GO_analyse(eSet=AlvMac, f="Treatment",
    subset=list(Time=c("48H"), Treatment=c("MB", "TB")))

## Not run:
# Syntax examples without actual data:

# To force the use of the Ensembl BioMart for the human species, use:
GO_analyse(eSet, f, biomart_dataset = "hsapiens_gene_ensembl")

# To force use of the bovine affy_bovine microarray annotations use:
GO_analyse(eSet, f, microarray = "affy_bovine")

# Valid Ensembl BioMart datasets are listed in the following variable
data(prefix2dataset)
```

```
# Valid microarray= values are listed in the following variable
data(microarray2dataset)


# Additional time-consuming example usages, can be run manually
# with test data:

# Run the analysis on factor "Treatment" including all samples
# Factor must be one of the column names in pData(targets)
AlvMac_results <- GO_analyse(eSet=AlvMac, f="Treatment")

# Run the analysis on factor "Treatment" using ANOVA method
AlvMac_results <- GO_analyse(eSet=AlvMac, f="Treatment", method="a")

## End(Not run)
```

`heatmap_GO` *Generates a heatmap and hierarchical clustering of the samples and the genes*

Description

Clusters the samples and the genes associated with a GO term using the expression levels of genes related to a given ontology. Represents expression levels of those genes in a heatmap.

Usage

```
heatmap_GO(
    go_id, result, eSet, f=result$factor, subset=NULL, gene_names=TRUE,
    scale="none", cexCol=1.2, cexRow=0.5,
    cex.main=1, trace="none", expr.col=bluered(75),
    row.col.palette="Accent",
    row.col=c(),
    main=paste(go_id, result$GO[result$GO$go_id == go_id,
                                 "name_1006"]),
    main.Lsplit=NULL,
    ...)
```

Arguments

go_id	A Gene Ontology (GO) identifier.
result	The output of `GO_analyse()` or a subset of it obtained from `subset_scores()`.
eSet	`ExpressionSet` of the `Biobase` package including a gene-by-sample expression matrix in the `assayData` slot, and a phenotypic information data-frame in the `phenodata` slot. In the expression matrix, row names are Ensembl gene identifiers or probeset identifiers, and column names are sample identifiers. In the phentypic data-frame, row names are sample idenfifiers, column names are grouping factors and phenotypic traits usable for the one-way ANOVA.

`f`	The grouping factor in `phenodata` to label the samples by.
`subset`	A named list to subset `eSet`. Names must be column names existing in colnames(pData(eSet)). Values must be vectors of values existing in the corresponding column of pData(eSet).
`gene_names`	A boolean value. Default is TRUE, to label genes by their associated gene name. If FALSE, labels the genes by their feature identifier in the expression dataset (i.e. Ensembl gene identifier or microarray probeset).
`scale`	Character indicating if the values should be centered and scaled in either the row direction or the column direction, or none. Default is "none". See `heatmap.2()`.
`cexCol, cexRow`	Positive numbers, used as cex.axis in for the row or column axis labeling. Defaults are 1.2 and 1, respectively. See `heatmap.2()`.
`cex.main`	Scaling factor of the main title font size. Default is 1. We suggest to use it in combination with the argument `main.Lsplit` for GO terms with long names.
`trace`	Character string indicating whether a solid "trace" line should be drawn across each 'row' or down each 'column', both' or 'none'. The distance of the line from the center of each color-cell is proportional to the size of the measurement. Defaults to 'none'.
`expr.col`	Character vector indicating the colors to represent the different levels of gene expression. Defaults to a colormap of 75 shades ranging from blue (low) to red (high) and centered around white. If using differential expression data, you should probably use `greenred(75)` instead.
`row.col.palette`	A valid `RColorBrewer` palette name to fetch the colormap from, to color-code the groups of samples.
`row.col`	A vector of color names or codes. The number of colors provided must match the number of levels of the grouping factor. Default to an palette of up to 9 colors marking the different levels of the predefined grouping factor on the left side of the heatmap.
`main`	Main title of the figure. Default is paste(go_id, go_name).
`main.Lsplit`	Number of characters after which a new-line character will be inserted in the main title. If this would occur within a word, the new-line character will be inserted before this word. Default is NULL, leaving the title on a single line.
`...`	Additional arguments passed on to `heatmap.2()`.

Value

Returns the output of the `heatmap.2()` function.

Author(s)

Kevin Rue-Albrecht

See Also

Method `heatmap.2`, `GO_analyse`, and `brewer.pal.info`.

Examples

```
# load the sample output data
data(AlvMac_results)

# Heatmap the top-ranked GO term (toll-like receptor 4 signaling pathway) as
# example
heatmap_GO(go_id="GO:0034142", result=AlvMac_results, eSet=AlvMac)

# Same with larger sample labels on the right hand side.
heatmap_GO(go_id="GO:0034142", result=AlvMac_results, eSet=AlvMac, cexRow=1)

# Change the color-coding to green-black-red gradient (more appropriate for
# differential expression values)
library(gplots)
heatmap_GO(go_id="GO:0034142", result=AlvMac_results, eSet=AlvMac,
    expr.col=greenred(75))
```

`hist_scores` *Plots the distribution of scores following an GOexpress analysis.*

Description

Plots the an histogram representing the frequencies of scores in the output variable of the GO_analyse() function.

This function can also be used on the output of `subset_scores()` function as it returns a value formatted identically to the output of the `GO_analyse()` function.

Usage

```
hist_scores(result,
            main=paste("Distribution of average scores in",
                       deparse(substitute(result))),
            xlab="Average score", ...)
```

Arguments

`result`	The output of the `GO_analyse()` function.
`main, xlab`	These arguments to title have useful defaults here.
`...`	Additional arguments passed on to `hist()`.

Value

Returns the output of the `hist()` function.

Author(s)

Kevin Rue-Albrecht

See Also

Method `hist`, and `GO_analyse`.

Examples

```
# load the sample output data
data(AlvMac_results)

# Histogram of scores (labelled with counts)
hist_scores(result=AlvMac_results, breaks=20, labels=TRUE)

# filter for Biological Processes associated with 10+ genes
filtered_results <- subset_scores(result=AlvMac_results, total_count=5,
    namespace="BP")

# Histogram of scores (labelled with counts)
hist_scores(result=filtered_results, breaks=20, labels=TRUE)
```

`list_genes` *Returns the genes associated with a Gene Ontology*

Description

Given a Gene Ontology (GO) identifier represented in the dataset, returns a character vector listing the feature identifiers annotated to it.

Usage

```
list_genes(go_id, result, data.only=TRUE)
```

Arguments

`go_id`	A Gene Ontology (GO) identifier represented in the dataset.
`result`	The output of `GO_analyse()` or a subset of it obtained from `subset_scores()`.
`data.only`	Whether to return only the feature identifiers present in the given dataset or alternatively returns all feature identifiers associated with the GO term in the Ensembl BioMart. Default is TRUE.

Value

A character vector listing the feature identifiers of the genes associated with the GO term.

Author(s)

Kevin Rue-Albrecht

See Also

Method GO_analyse.

Examples

```
# load the sample output data
data(AlvMac_results)

# List of genes associated with the GO term "toll-like receptor 4 signaling
# pathway"
list_genes(result=AlvMac_results, go_id="GO:0034142")
```

---

microarray2dataset *Table mapping probeset identifier prefixes to datasets in the Ensembl BioMart.*

Description

The species corresponding to an probeset identifier can often be identified from the prefix of the identifier (e.g. Bt.457.1.S1_at corresponds to *Bos taurus*). This table maps some known unique prefix to the corresponding species.

Usage

```
data(microarray2dataset)
```

Details

All Agilent microarray share the same prefix pattern, making it very difficult to differentiate. Many Affymetrix microarrays also share the same prefix pattern for several probesets.

The `dataset` and `microarray` arguments of the `GO_analyse` method are the best way to specify which BioMart information should be used to annotate the features in your expression dataset.

Value

A data frame with 105 observations of the 5 variables described below:

- `dataset` contains species-specific biomaRt dataset names.
- `microarray` contains microarray identifiers in the Ensembl BioMart dataset described above.
- `sample` contains an example probeset sampled from the corresponding Ensembl BioMart dataset and microarray.
- `prefix` a manually curated prefix pattern for the corresponding Ensembl BioMart dataset and microarray. This pattern may or may not be unique to the microarray in the BioMart database. Therefore we encourage users to use the 'microarray' .....
- `unique` contains a boolean value stating whether the prefix pattern was found curated as unique to the microarray or not.

Source

A separate script was used to query the Ensembl BioMart server and build this table.

Examples

```
data(microarray2dataset)
microarray2dataset
```

`overlap_GO` *Shared genes between a list of GO terms.*

Description

Given a list of two to five GO terms, overlap_GO() will produce a Venn diagram showing the counts of overlapping genes associated with those GO terms.

Usage

```
overlap_GO(go_ids, result, filename=NULL, mar=rep(0.1, 4), ...)
```

Arguments

`go_ids`	A character vector of GO term identifiers to compare. For instance, `head(result$scores$go_id, n=5)` to compare the first 5 top- ranked GO terms in the`result` variable.
`result`	The output of `GO_analyse()` or a subset of it obtained from `subset_scores()`.
`filename`	The output filename where the Venn diagram will be saved. Default is NULL, displaying the Venn diagram on screen.
`mar`	The margins around the Venn diagram. Some Venn diagrams place the GO term labels outside the visible frame.
`...`	Further parameters forwarded to `the venn.diagram()` function.

Value

Returns the output of the `venn.diagram()` function.

Warning

An error is returned if the list of GO term identifiers contains less than 2 elements or more than 5, as the underlying `venn.diagram()` method does not support values outside that range.

Author(s)

Kevin Rue-Albrecht

See Also

Method `venn.diagram`.

Examples

```
# load the sample output data
data(AlvMac_results)

# filter for Biological Processes associated with 10+ genes
filtered_results <- subset_scores(result=AlvMac_results, total_count=5,
    namespace="BP")

# Venn diagram of overlapping genes between top 5 GO terms
overlap_GO(go_ids=head(filtered_results$GO$go_id, n=5),
    result=filtered_results, filename="VennDiagram.tiff")
```

---

plot_design *Plot Univariate Effects for genes associated with a Gene Ontology*

Description

Successively plots univariate effects of one or more `factors`, typically for a designed experiment as analyzed by `aov`().

Usage

```
plot_design(
    go_id, result, eSet,
    factors=colnames(pData(eSet)), main="", main.Lsplit=NULL, ...)
```

Arguments

`go_id`	A Gene Ontology (GO) identifier represented by at least one gene in the dataset.
`result`	The output of `GO_analyse()` or a subset of it obtained from `subset_scores()`.
`eSet`	`ExpressionSet` of the `Biobase` package including a gene-by-sample expression matrix in the `AssayData` slot, and a phenotypic information data-frame in the `phenodata` slot. In the expression matrix, row names are Ensembl gene identifiers or probeset identifiers, and column names are sample identifiers. In the phentypic data-frame, row names are sample idenfifiers, column names are grouping factors and phenotypic traits usable for the one-way ANOVA.
`factors`	A set of column names from `phenodata`. Each of these values will be represented on the X-axis to investigate its effect on the average expression of a given genes for each level of that factor.
`main`	Changes the main title of the plots.
`main.Lsplit`	Number of characters after which a new-line character will be inserted in the main title. If this would occur within a word, the new-line character will be inserted before this word. Default is NULL, leaving the title on a single line.
`...`	Additional arguments which will be passed on to the `plot.design`() function.

Value

The output of the `plot.design()` function.

Author(s)

Kevin Rue-Albrecht

See Also

Method `plot.design`.

Examples

```
# load the sample output data
data(AlvMac_results)

# Univariate plot
plot_design(go_id="GO:0034142", eSet=AlvMac, result=AlvMac_results)
```

`prefix2dataset` *Table mapping Ensembl gene identifier prefixes to BioMart datasets.*

Description

The species corresponding to an Ensembl gene identifier can typically be identified from the prefix of the identifier (e.g. ENSBTAG corresponds to *Bos taurus*). This table maps each known unique prefix to the corresponding species.

Usage

```
data(prefix2dataset)
```

Details

C. elegans, *D. melanogaster*, and *S. cerevisiae* have atypical identifier pattern and prefixes. Prefixes for those are included in the table, but checked independently of the other species.

Value

A data frame with 66 rows and 2 columns. The columns are described below:

- `prefix` contains unique prefixes.
- `dataset` contains the corresponding biomaRt dataset name.

Source

A separate script was used to query the Ensembl BioMart server and build this table.

Examples

```
data(prefix2dataset)
str(prefix2dataset)
prefix2dataset
```

---

quantiles_scores — *Returns the quantiles of scores following an GOexpress analysis.*

Description

Returns a set of quantiles in indicating the scores reached by given proportions of the GO terms.

This function can also be used on the output of `subset_scores()` function as it returns a value formatted identically to the output of the `GO_analyse()` function.

Usage

```
quantiles_scores(result, probs=c(0.9, 0.95, 0.99, 0.999, 0.9999),
    quartiles=FALSE)
```

Arguments

`result`	The output of `GO_analyse()` or a subset of it obtained from `subset_scores()`.
`probs`	Numeric vector of probabilities with values in [0,1]. (Values up to 2e-14 outside that range are accepted and moved to the nearby endpoint.) See `quantile`
`quartiles`	A numeric vector of the percentiles for which the scores are desired.

Value

A named vector of percentiles and corresponding scores.

Author(s)

Kevin Rue-Albrecht

See Also

Method `quantile`.

Examples

```
# load the sample output data
data(AlvMac_results)

# filter for Biological Processes associated with 10+ genes
filtered_results <- subset_scores(result=AlvMac_results, total_count=5,
    namespace="BP")

# Quantiles of scores
quantiles_scores(result=filtered_results)
```

rerank *Reorder the result variable by alternative metrics.*

Description

Reorder the ranked tables of GO terms and genes either by increasing (average) rank or decreasing (average) score.

Usage

```
rerank(result, rank.by = "rank")
```

Arguments

`result`	The output of `GO_analyse()` or a subset of it obtained from `subset_scores()`.
`rank.by`	Either of "rank" or "score"; the metric to rank the GO terms and genes by.

Value

A list formatted identically to the results of the analysis, but ordered by the chosen metric.

Note

The name reorder() was not used to avoid conflict with package `stats`.

Author(s)

Kevin Rue-Albrecht

See Also

Method `GO_analyse`.

Examples

```
# load the sample output data
data(AlvMac_results)

# Re-rank the GO terms and genes based on the actual score instead of the rank
reranked.byScore <- rerank(result=AlvMac_results, rank.by="score")
```

subEset *Subset an ExpressionSet to values of phenotypic data columns.*

Description

Given a list of column names and corresponding values present in the phenoData slot of an ExpressionSet object, this method returns the subset of the ExpressionSet restricted to samples associated with the given values in the given columns.

Usage

```
subEset(eSet, subset = list())
```

Arguments

`eSet`	`ExpressionSet` of the `Biobase` package including a gene-by-sample expression matrix in the `AssayData` slot, and a phenotypic information data-frame in the `phenodate` slot. In the expression matrix, row names are Ensembl gene identifiers or probeset identifiers, and column names are sample identifiers. In the phentypic data-frame, row names are sample idenfifiers, column names are grouping factors and phenotypic traits usable for the one-way ANOVA.
`subset`	A named list to subset `eSet`. Names must be column names existing in colnames(pData(eSet)). Values must be vectors of values existing in the corresponding column of pData(eSet).

Value

A subset of the given ExpressionSet restricted to samples associated with the given values in the given columns

Author(s)

Kevin Rue-Albrecht

See Also

Class `ExpressionSet`.

Examples

```
# Load example data set
data(AlvMac)

# Subset it to only samples of "CN" and "MB" treatments, and also only "2H",
# "6H", and "24H" time-points
sub.AlvMac <- subEset(eSet=AlvMac, subset=list(Treatment=c("CN","MB"),
                        Time=c("2H","6H")))
```

subset_scores — *Returns a filtered list from GO_analyse results.*

Description

Given an output variable from a `GO_analyse` analysis and a set of valid filters and thresholds, returns an identically formatted list keeping only the rows of the score table passing all the filters.

Usage

```
subset_scores(result, ...)

# Suggested use:
# subset_scores(result, total_count=5, namespace_1003="biological_process")
# synonym to
# subset_scores(result, total=5, namespace="BP")
```

Arguments

`result`	The output of the `GO_analyse()` function.
`...`	Additional pairs of filter and threshold values in the format "filter=threshold". Filters must be valid names from colnames(result$scores).

Details

It is critical to filter out GO terms with very few genes (e.g. less than 5 genes), as the scoring function is biased for those GO terms (see UsersGuide).

In addition, it is useful to retain only GO terms of one type (i.e. namespace) among the three possible: "biological process", "molecular function", and "cellular component".

Suggested filters:

total_count, total:	Filter on the number of genes annotated to the GO term.
namespace, namespace_1003:	Filter keeping only the GO terms of a given type. Valid values are "biological_process", "mol

Other filters:

data_count, data:	Filter on the number of genes in the dataset annotated to the GO term.
ave_rank:	Average of the rank of all genes annotated to the GO term. Genes annotated to the GO term but absent fron
ave_score:	Average of the score of all genes annotated to the GO term. Scores are the mean decrease in Gini index for

Value

A list formatted identically to the results of the analysis, but restricted to the gene ontologies passing the given filters, and the genes mapped to those ontologies.

Author(s)

Kevin Rue-Albrecht

See Also

Method `GO_analyse`.

Examples

```
# load the sample output data
data(AlvMac_results)

# have an overview of the result variable
str(AlvMac_results)

# filter for Biological Processes associated with 10+ genes
filtered_results <- subset_scores(result=AlvMac_results, total_count=5,
    namespace="BP")

# have an overview of the filtered result variable
str(filtered_results)
```

`table_genes` *Returns a table listing the genes associated with a given Gene Ontology*

Description

Given a Gene Ontology (GO) identifier represented in the dataset and the output variable of a `GO_analyse()` function, `table_genes()` returns a table listing the genes associated with that go_id, their associated gene name, and description.

Usage

```
table_genes(go_id, result, data.only=FALSE)
```

Arguments

`go_id`	A Gene Ontology (GO) identifier.
`result`	The output of the `GO_analyse()` function.
`data.only`	Whether to return only the feature identifiers present in the given dataset or alternatively returns all feature identifiers associated with the GO term in the Ensembl BioMart.

Value

A data frame listing the statistics and annotations for the genes present in the expression dataset and associated with the GO term.

Author(s)

Kevin Rue-Albrecht

See Also

Method `GO_analyse`.

Examples

```
# load the sample output data
data(AlvMac_results)

# Table of result for genes associated with the GO term
# "toll-like receptor 4 signaling pathway"
table_genes(result=AlvMac_results, go_id="GO:0034142")
```

Index