Package 'EnrichmentBrowser'

April 9, 2015

Version 1.0.3

Date 2015-02-20

Title Seamless navigation through combined results of set-based and network-based enrichment analysis

Author Ludwig Geistlinger

Maintainer Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

Depends R(>= 3.0.0), Biobase, KEGGgraph

Imports KEGGREST, MASS, RColorBrewer, Rgraphviz, SPIA, graph, limma, mixtools, neaGUI, qvalue, safe, stringr

Suggests ALL, BiocStyle, hgu95av2.db

Description The EnrichmentBrowser package implements essential functionality for the enrichment analysis of gene expression data. The analysis combines the advantages of set-based and network-based enrichment analysis in order to derive high-confidence gene sets and biological pathways that are differentially regulated in the expression data under investigation. In addition, the package facilitates the visualization and exploration of such sets and pathways.

License Artistic-2.0

biocViews Microarray, GeneExpression, DifferentialExpression, Pathways, GraphAndNetwork, Network, GeneSetEnrichment, NetworkEnrichment, Visualization, ReportWriting

R topics documented:

`comb.ea.results` *Combining enrichment analysis results*

Description

Different enrichment analysis methods usually result in different gene set rankings for the same dataset. This function allows to combine results from the different set-based and network-based enrichment analysis methods. This includes the computation of average gene set ranks across methods and of combined gene set p-values.

Usage

```
comb.ea.results(res.list, pcomb.meth=c("fisher","stouffer"), out.file=NULL)
```

Arguments

`res.list`	A list of enrichment analysis result objects as returned by the functions ‘sbea’ and ‘nbea’.
`pcomb.meth`	P-value combination method. See details. Defaults to ‘fisher’.
`out.file`	Optional output file the combined ranking is written to.

Details

Fisher’s method (pcomb.meth="fisher") combines the statistical significance from several independent tests. Stouffer’s method (pcomb.meth="stouffer") allows the combination from dependent tests. See Kim et al., 2013.

Value

if(is.null(out.file)): an enrichment analysis result object that can be detailedly explored by calling ’ea.browse’ and from which a flat gene set ranking can be extracted by calling ’gs.ranking’. If ’out.file’ is given, the ranking is written to the specified file.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

References

Kim et al. (2013) Stouffer's test in a large scale simultaneous hypothesis testing PLoS One, 8(5), e63290.

See Also

sbea, nbea, ea.browse

Examples

```
# (1) reading the expression data from file
exprs.file <- system.file("extdata/ALL_exprs.tab", package="EnrichmentBrowser")
pdat.file <- system.file("extdata/ALL_pData.tab", package="EnrichmentBrowser")
fdat.file <- system.file("extdata/ALL_fData.tab", package="EnrichmentBrowser")
probe.eset <- read.eset(exprs.file, pdat.file, fdat.file)

# (2) summarizing probe expression on gene level
gene.eset <- probe.2.gene.eset(probe.eset)

# (3a) getting all human KEGG gene sets
# hsa.gs <- get.kegg.genesets("hsa")
gs.file <- system.file("extdata/hsa_kegg_gs.gmt", package="EnrichmentBrowser")
hsa.gs <- parse.genesets.from.GMT(gs.file)

# (3b) compiling gene regulatory network from KEGG pathways
# hsa.grn <- compile.grn.from.kegg("hsa")
pwys <- system.file("extdata/hsa_kegg_pwys.zip", package="EnrichmentBrowser")
hsa.grn <- compile.grn.from.kegg(pwys)

# (4) performing the set-based and network-based enrichment analysis
# Note: reduced permutations for demonstration
#       recommended default is 1000 permutations
# sbea.res <- sbea(method="gsea", eset=gene.eset, gs=hsa.gs)
# nbea.res <- nbea(method="ggea", eset=gene.eset, gs=hsa.gs, grn=hsa.grn)
sbea.res <- sbea(method="ora", eset=gene.eset, gs=hsa.gs, perm=0)
nbea.res <- nbea(method="ggea",
                    eset=gene.eset, gs=hsa.gs, grn=hsa.grn, perm=100)

# (5) combining the results
res.list <- list(sbea.res, nbea.res)
comb.res <- comb.ea.results(res.list)

# (6) result visualization and exploration
gs.ranking(comb.res)

ea.browse(comb.res)
```

`compile.grn.from.kegg` *Compilation of a gene regulatory network from KEGG pathways*

Description

To perform network-based enrichment analysis a gene regulatory network (GRN) is required. There are well-studied processes and organisms for which comprehensive and well-annotated regulatory networks are available, e.g. the RegulonDB for E. coli and Yeastract for S. cerevisiae. However, in many cases such a network is missing. A first simple workaround is to compile a network from regulations in the KEGG database.

Usage

```
compile.grn.from.kegg(pwys, out.file = NULL)
```

Arguments

`pwys`	Either a list of KEGGPathway objects or an absolute file path of a zip compressed archive of pathway xml files in KGML format. Alternatively, you can specify an organism in KEGG three letter code, e.g. 'hsa' for 'homo sapiens', and the pathways will be downloaded automatically.
`out.file`	Optional output file the gene regulatory network will be written to.

Value

if(is.null(out.file)): the gene regulatory network else: none, as the gene regulatory network is written to file

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

See Also

`KEGGPathway-class`, `parseKGML`, `download.kegg.pathways`

Examples

```
# (1) download human pathways
#   pwys <- download.kegg.pathways("hsa")
# (2) compile gene regulatory network
#   grn <- compile.grn.from.kegg(pwys)

pwys <- system.file("extdata/hsa_kegg_pwys.zip", package="EnrichmentBrowser")
hsa.grn <- compile.grn.from.kegg(pwys)
```

`de.ana` *Differential expression analysis between two sample groups*

Description

The function carries out a differential expression analysis between two sample groups. Resulting fold changes and t-test derived p-values for each feature are appended to the fData slot. Raw p-values are corrected for multiple testing using the method from Benjamini and Hochberg.

Usage

```
de.ana(eset)
```

Arguments

`eset`	An object of class 'ExpressionSet'

Value

An object of `ExpressionSet-class` with measures of differential expression annotated in the fData slot.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

See Also

`ExpressionSet-class`, `eBayes`, `p.adjust`

Examples

```
# reading the expression data from file
exprs.file <- system.file("extdata/ALL_exprs.tab", package="EnrichmentBrowser")
pdat.file <- system.file("extdata/ALL_pData.tab", package="EnrichmentBrowser")
fdat.file <- system.file("extdata/ALL_fData.tab", package="EnrichmentBrowser")
eset <- read.eset(exprs.file, pdat.file, fdat.file)
eset <- de.ana(eset)
head(fData(eset))
```

`download.kegg.pathways` *Download of KEGG pathways for a particular organism*

Description

The function downloads all metabolic and non-metabolic pathways in KEGG XML format for a specified organism.

Usage

```
download.kegg.pathways(org, out.dir = NULL, zip = FALSE)
```

Arguments

`org`	Organism in KEGG three letter code, e.g. 'hsa' for 'homo sapiens'.
`out.dir`	Output directory. If not null, pathways are written to files in the specified directory.
`zip`	Logical. In case pathways are written to file ('out.dir' is not null): should output files be zipped?

Value

if(is.null(out.dir)): a list of KEGGPathway objects else: none, as pathways are written to file

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

See Also

`keggList`, `keggGet`, `KEGGPathway-class`, `parseKGML`

Examples

```
    pwys <- download.kegg.pathways("hsa")
```

`ea.browse` *Exploration of enrichment analysis results*

Description

Functions to extract a flat gene set ranking from an enrichment analysis result object and to detailedly explored it.

Usage

```
ea.browse(res, nr.show=-1, set.view=TRUE, graph.view=NULL)

gs.ranking(res, signif.only=TRUE)
```

Arguments

`res`	Enrichment analysis result object as returned by the functions 'sbea' and 'nbea'.
`nr.show`	Number of gene sets to show. As default all statistical significant gene sets are displayed.
`set.view`	Logical. Should a set-based summary (includes a report, plots, and browsing in KEGG) be created for the result? Defaults to TRUE.
`graph.view`	Optional. Should a graph-based summary (reports and visualizes consistency of regulations) be created for the result? If specified, it needs to be a gene regulatory network, i.e. either an absolute file path to a tabular file or a character matrix with exactly *THREE* cols; 1st col = IDs of regulating genes; 2nd col = corresponding regulated genes; 3rd col = regulation effect; Use '+' and '-' for activation/inhibition.
`signif.only`	Logical. Display only those gene sets in the ranking which satisfy the significance level? Defaults to TRUE.

Value

gs.ranking: data.frame with gene sets ranked by the corresponding p-value;

ea.browse: none, opens the browser to explore results.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

See Also

`sbea`, `nbea`, `comb.ea.results`

Examples

```
# (1) reading the expression data from file
exprs.file <- system.file("extdata/ALL_exprs.tab", package="EnrichmentBrowser")
pdat.file <- system.file("extdata/ALL_pData.tab", package="EnrichmentBrowser")
fdat.file <- system.file("extdata/ALL_fData.tab", package="EnrichmentBrowser")
probe.eset <- read.eset(exprs.file, pdat.file, fdat.file)

# (2) summarizing probe expression on gene level
gene.eset <- probe.2.gene.eset(probe.eset)

# (3) getting all human KEGG gene sets
# hsa.gs <- get.kegg.genesets("hsa")
gs.file <- system.file("extdata/hsa_kegg_gs.gmt", package="EnrichmentBrowser")
hsa.gs <- parse.genesets.from.GMT(gs.file)

# (4) performing the enrichment analysis
ea.res <- sbea(method="ora", eset=gene.eset, gs=hsa.gs, perm=0)

# (5) result visualization and exploration
gs.ranking(ea.res)
ea.browse(ea.res)
```

`ebrowser` *Seamless navigation through enrichment analysis results*

Description

This is the all-in-one wrapper function to perform the standard enrichment analysis pipeline implemented in the EnrichmentBrowser package.

Given flat gene expression data, the data is read in and subsequently subjected to chosen enrichment analysis methods.

The results from different methods can be combined and investigated in detail in the default browser.

Usage

```
ebrowser( meth, exprs, pdat, fdat,
        gs, grn=NULL, perm=1000, alpha=0.05, beta=1,
        comb=FALSE, browse=TRUE, nr.show=-1, out.dir=NULL )
```

Arguments

`meth`	Enrichment analysis method. Currently, the following enrichment analysis methods are supported: ‘ora’, ‘safe’, ‘gsea’, ‘samgs’, ‘ggea’, ‘spia’, and ‘nea’. See ‘sbea’ and ‘nbea’ for details.
`exprs`	Expression matrix. A tab separated text file containing *normalized* expression values on a *log* scale. Columns = samples/subjects; rows = features/probes/genes; NO headers, row or column names. Supported data types are log2 counts (microarray single-channel), log2 ratios (microarray two-color), and log2-counts

	per million (RNA-seq logCPMs). See the limma's user guide http://www.bioconductor.org/packages/relea for definition and normalization of the different data types.
`pdat`	Phenotype data. A tab separated text file containing annotation information for the samples in either *two or three* columns. NO headers, row or column names. The number of rows/samples in this file should match the number of columns/samples of the expression matrix. The 1st colum is reserved for the sample IDs; The 2nd column is reserved for a *BINARY* group assignment. Use '0' and '1' for unaffected (controls) and affected (cases) sample class, respectively. For paired samples or sample blocks a third column is expected that defines the blocks.
`fdat`	Feature data. A tab separated text file containing annotation information for the features. Exactly *TWO* columns; 1st col = feature IDs; 2nd col = corresponding KEGG gene ID for each feature ID in 1st col; NO headers, row or column names. The number of rows/features in this file should match the number of rows/features of the expression matrix.
`gs`	Gene sets. Either a list of gene sets (vectors of KEGG gene IDs) or a text file in GMT format storing all gene sets under investigation.
`grn`	Gene regulatory network. Either an absolute file path to a tabular file or a character matrix with exactly *THREE* cols; 1st col = IDs of regulating genes; 2nd col = corresponding regulated genes; 3rd col = regulation effect; Use '+' and '-' for activation/inhibition.
`perm`	Number of permutations of the expression matrix to estimate the null distribution. Defaults to 1000.
`alpha`	Statistical significance level. Defaults to 0.05.
`beta`	Log2 fold change signifcance level. Defaults to 1 (2-fold).
`comb`	Logical. Should results be combined if more then one enrichment method is selected? Defaults to FALSE.
`browse`	Logical. Should results be displayed in the browser for interactive exploration? Defaults to TRUE.
`nr.show`	Number of gene sets to show. As default all statistical significant gene sets are displayed.
`out.dir`	Optional output directory all results will be written to.

Value

None, opens the browser to explore results.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

See Also

`read.eset` to read expression data from file; `probe.2.gene.eset` to transform probe to gene level expression; `get.kegg.genesets` to retrieve gene set definitions from KEGG; `compile.grn.from.kegg` to construct a GRN from KEGG pathways; `sbea` to perform set-based enrichment analysis; `nbea` to perform network-based enrichment analysis; `comb.ea.results` to combine results from different methods; `ea.browse` for exploration of resulting gene sets

Examples

```
# expression data from file
exprs.file <- system.file("extdata/ALL_exprs.tab", package="EnrichmentBrowser")
pdat.file <- system.file("extdata/ALL_pData.tab", package="EnrichmentBrowser")
fdat.file <- system.file("extdata/ALL_fData.tab", package="EnrichmentBrowser")

# getting all human KEGG gene sets
# hsa.gs <- get.kegg.genesets("hsa")
gs.file <- system.file("extdata/hsa_kegg_gs.gmt", package="EnrichmentBrowser")
hsa.gs <- parse.genesets.from.GMT(gs.file)


# set-based enrichment analysis
ebrowser(   meth="ora",
        exprs=exprs.file,
        pdat=pdat.file,
        fdat=fdat.file,
        gs=hsa.gs)


# compile a gene regulatory network from KEGG pathways
# hsa.grn <- compile.grn.from.kegg("hsa")
pwys <- system.file("extdata/hsa_kegg_pwys.zip", package="EnrichmentBrowser")
hsa.grn <- compile.grn.from.kegg(pwys)


# network-based enrichment analysis
ebrowser(   meth="ggea",
        exprs=exprs.file,
        pdat=pdat.file,
        fdat=fdat.file,
        gs=hsa.gs,
        grn=hsa.grn)


# combining results
ebrowser(   meth=c("ora", "ggea"),
        perm=100,
        exprs=exprs.file,
        pdat=pdat.file,
        fdat=fdat.file,
        gs=hsa.gs,
        grn=hsa.grn)
```

`get.kegg.genesets` *Definition of gene sets according to KEGG pathways for a specified organism*

Description

To perform a gene set enrichment analysis on KEGG pathways, it is necessary to build up the gene set database in a format that the GSEA method can read. Parsing a list of gene sets from a flat text file in GMT format. This function performs the necessary steps, including the retrieval of the participating gene IDs for each pathway and the conversion to GMT format.

Usage

```
get.kegg.genesets(pwys, gmt.file = NULL)

parse.genesets.from.GMT(gmt.file)
```

Arguments

`pwys`	Either a list of KEGGPathway objects or an absolute file path of a zip compressed archive of pathway xml files in KGML format. Alternatively, an organism in KEGG three letter code, e.g. ‘hsa’ for ‘homo sapiens’.
`gmt.file`	Gene set file in GMT format. See details.

Details

The GMT (Gene Matrix Transposed) file format is a tab delimited file format that describes gene sets. In the GMT format, each row represents a gene set. Each gene set is described by a name, a description, and the genes in the gene set. See references.

Value

A list of gene sets (vectors of gene IDs).

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

References

http://www.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats

See Also

`keggList`, `keggLink`, `KEGGPathway-class`, `parseKGML`

Examples

```
# WAYS TO DEFINE GENE SETS ACCORDING TO HUMAN KEGG PATHWAYS

# (1) from scratch: via organism ID

gs <- get.kegg.genesets("hsa")
```

```
# (2) extract from pathways
# download human pathways via:
# pwys <- download.kegg.pathways("hsa")
pwys <- system.file("extdata/hsa_kegg_pwys.zip", package="EnrichmentBrowser")
gs <- get.kegg.genesets(pwys)

# (3) parsing gene sets from GMT
gmt.file <- system.file("extdata/hsa_kegg_gs.gmt", package="EnrichmentBrowser")
gs <- parse.genesets.from.GMT(gmt.file)
```

`ggea.graph` *GGEA graphs of consistency between regulation and expression*

Description

Gene graph enrichment analysis (GGEA) is a network-based enrichment analysis method implemented in the EnrichmentBrowser package. The idea of GGEA is to evaluate the consistency of known regulatory interactions with the observed gene expression data. A GGEA graph for a gene set of interest displays the consistency of each interaction in the network that involves a gene set member. Nodes (genes) are colored according to expression (up-/down-regulated) and edges (interactions) are colored according to consistency, i.e. how well the interaction type (activation/inhibition) is reflected in the correlation of the expression of both interaction partners.

Usage

```
ggea.graph(gs, grn, eset, org=NA, alpha=0.05, beta=1, max.edges=50, cons.thresh=0.7)
 ggea.graph.legend()
```

Arguments

`gs`	Gene set under investigation. This should be a character vector of KEGG gene IDs.
`grn`	Gene regulatory network. Character matrix with exactly *THREE* cols; 1st col = IDs of regulating genes; 2nd col = corresponding regulated genes; 3rd col = regulation effect; Use '+' and '-' for activation/inhibition.
`eset`	Expression set. An object of class 'ExpressionSet' containing the gene expression set. See 'read.eset' and 'probe.2.gene.eset' for required annotations in the pData and fData slot.
`org`	Organism under investigation in KEGG three letter code, e.g. 'hsa' for 'homo sapiens'. Used to map gene IDs to gene symbols as displayed in KEGG pathway maps. If not annotated, gene IDs are plotted as default.
`alpha`	Statistical significance level. Defaults to 0.05.
`beta`	Log2 fold change significance level. Defaults to 1 (2-fold).
`max.edges`	Maximum number of edges that should be displayed. Defaults to 50.
`cons.thresh`	Consistency threshold. Graphical parameter that correspondingly increases line width of edges with a consistency above the chosen threshold (defaults to 0.7).

Value

None, plots to a graphics device.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

See Also

nbea to perform network-based enrichment analysis. ea.browse for exploration of resulting gene sets.

Examples

```
# (1) reading the expression data from file
exprs.file <- system.file("extdata/ALL_exprs.tab", package="EnrichmentBrowser")
pdat.file <- system.file("extdata/ALL_pData.tab", package="EnrichmentBrowser")
fdat.file <- system.file("extdata/ALL_fData.tab", package="EnrichmentBrowser")
probe.eset <- read.eset(exprs.file, pdat.file, fdat.file)

# (2) summarizing probe expression on gene level
gene.eset <- probe.2.gene.eset(probe.eset)

# (3a) getting all human KEGG gene sets
# hsa.gs <- get.kegg.genesets("hsa")
gs.file <- system.file("extdata/hsa_kegg_gs.gmt", package="EnrichmentBrowser")
hsa.gs <- parse.genesets.from.GMT(gs.file)

# (3b) compiling gene regulatory network from KEGG pathways
# hsa.grn <- compile.grn.from.kegg("hsa")
pwys <- system.file("extdata/hsa_kegg_pwys.zip", package="EnrichmentBrowser")
hsa.grn <- compile.grn.from.kegg(pwys)

# plot consistency graph
ggea.graph(gs=hsa.gs[["hsa05416_Viral_myocarditis"]],
                                            grn=hsa.grn, eset=gene.eset)

# get legend
ggea.graph.legend()
```

---

nbea *Network-based enrichment analysis (NBEA)*

Description

This is the main function for network-based enrichment analysis. It implements and uses existing implementations of several frequently used state-of-art methods and allows a flexible inspection of resulting gene set rankings.

Usage

```
nbea(method=c("ggea", "nea", "spia"), eset, gs, grn,
                alpha=0.05, perm=1000, out.file=NULL, browse=FALSE, ...)

nbea.methods()
```

Arguments

`method`	Network-based enrichment analysis method. Currently, the following network-based enrichment analysis methods are supported: ‘ggea’, ‘nea’, ‘spia’. See Details. Default is ’ggea’. This can also be the name of a tailored function implementing network-based enrichment. See Details.
`eset`	Expression set. Either an object of class ’ExpressionSet’ or an absolute file path to an RData file containing the gene expression set. See ’read.eset’ and ’probe.2.gene.eset’ for required annotations in the pData and fData slot.
`gs`	Gene sets. Either a list of gene sets (vectors of KEGG gene IDs) or a text file in GMT format storing all gene sets under investigation.
`grn`	Gene regulatory network. Either an absolute file path to a tabular file or a character matrix with exactly *THREE* cols; 1st col = IDs of regulating genes; 2nd col = corresponding regulated genes; 3rd col = regulation effect; Use ’+’ and ’-’ for activation/inhibition.
`alpha`	Statistical significance level. Defaults to 0.05.
`perm`	Number of permutations of the expression matrix to estimate the null distribution. Defaults to 1000. If using method=‘ggea’, it is possible to set perm < 1 to use a fast approximation of gene set significance to avoid permutation testing. See Details.
`out.file`	Optional output file the gene set ranking will be written to.
`browse`	Logical. Should results be displayed in the browser for interactive exploration? Defaults to FALSE.
`...`	Additional arguments passed to individual nbea methods. This includes currently for GGEA: • beta: Log2 fold change significance level. Defaults to 1 (2-fold). • cons.thresh: consistency threshold. Defaults to -1. • gs.edges: Decides which edges of the grn are considered for a gene set under investigation. Should be one out of c(’&’, ’|’), denoting logical AND and OR. respectively. Accordingly, this either includes edges for which regulator AND / OR target gene are members of the investigated gene set.

Details

’ggea’: gene graph enrichment analysis, scores gene sets according to consistency within the given gene regulatory network, i.e. checks activating regulations for positive correlation and repressing regulations for negative correlation of regulator and target gene expression (Geistlinger et al., 2011). When using ’ggea’ it is possible to estimate the statistical significance of the consistency score of each gene set in two different ways: (1) based on sample permutation as described in the original publication (Geistlinger et al., 2011) or (2) using an approximation based on Bioconductor's npGSEA package that is much faster.

'nea': network enrichment analysis, implemented in Bioconductor's neaGUI package.

'spia': signaling pathway impact analysis, implemented in Bioconductor's SPIA package.

It is also possible to use additional network-based enrichment methods. This requires to implement a function that takes 'eset', 'gs', 'grn', 'alpha', and 'perm' as arguments and returns a numeric matrix 'res.tbl' with a mandatory column named 'P.VALUE' storing the resulting p-value for each gene set in 'gs'. The rows of this matrix must be named accordingly (i.e. rownames(res.tbl) == names(gs)). See examples.

Value

nbea.methods: a character vector of currently supported methods;

nbea: if(is.null(out.file)): an enrichment analysis result object that can be detailedly explored by calling 'ea.browse' and from which a flat gene set ranking can be extracted by calling 'gs.ranking'. If 'out.file' is given, the ranking is written to the specified file.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

References

Geistlinger et al. (2011) From sets to graphs: towards a realistic enrichment analysis of transcriptomic systems. Bioinformatics, 27(13), i366–73.

See Also

Input: `read.eset`, `probe.2.gene.eset` `get.kegg.genesets` to retrieve gene set definitions from KEGG. `compile.grn.from.kegg` to construct a GRN from KEGG pathways.

Output: `gs.ranking` to rank the list of gene sets. `ea.browse` for exploration of resulting gene sets.

Other: `sbea` to perform set-based enrichment analysis. `comb.ea.results` to combine results from different methods. `spia` for more information on signaling pathway impact analysis. `nea` for more information on network enrichment analysis.

Examples

```
# currently supported methods
nbea.methods()

# (1) reading the expression data from file
exprs.file <- system.file("extdata/ALL_exprs.tab", package="EnrichmentBrowser")
pdat.file <- system.file("extdata/ALL_pData.tab", package="EnrichmentBrowser")
fdat.file <- system.file("extdata/ALL_fData.tab", package="EnrichmentBrowser")
probe.eset <- read.eset(exprs.file, pdat.file, fdat.file)

# (2) summarizing probe expression on gene level
gene.eset <- probe.2.gene.eset(probe.eset)
```

```
# (3a) getting all human KEGG gene sets
# hsa.gs <- get.kegg.genesets("hsa")
gs.file <- system.file("extdata/hsa_kegg_gs.gmt", package="EnrichmentBrowser")
hsa.gs <- parse.genesets.from.GMT(gs.file)

# (3b) compiling gene regulatory network from KEGG pathways
# hsa.grn <- compile.grn.from.kegg("hsa")
pwys <- system.file("extdata/hsa_kegg_pwys.zip", package="EnrichmentBrowser")
hsa.grn <- compile.grn.from.kegg(pwys)

# (4) performing the enrichment analysis
# Note: reduced permutations for demonstration
#       recommended default is 1000 permutations
# ea.res <- nbea(method="ggea", eset=gene.eset, gs=hsa.gs, grn=hsa.grn)
ea.res <- nbea(method="ggea",
                eset=gene.eset, gs=hsa.gs, grn=hsa.grn, perm=100)

# (5) result visualization and exploration
gs.ranking(ea.res)

ea.browse(ea.res, graph.view=hsa.grn)


# using your own tailored function as enrichment method
dummy.nbea <- function(eset, gs, grn, alpha, perm)
{
    sig.ps <- sample(seq(0,0.05, length=1000),5)
    insig.ps <- sample(seq(0.1,1, length=1000), length(gs)-5)
    ps <- sample(c(sig.ps, insig.ps), length(gs))
    score <- sample(1:100, length(gs), replace=TRUE)
    res.tbl <- cbind(score, ps)
    colnames(res.tbl) <- c("SCORE", "P.VALUE")
    rownames(res.tbl) <- names(gs)
    return(res.tbl[order(ps),])
}

nbea.res2 <- nbea(method="dummy.nbea",
    eset=gene.eset, gs=hsa.gs, grn=hsa.grn)
gs.ranking(nbea.res2)
```

plots *Visualization of gene expression*

Description

Visualization of differential gene expression via heatmaps, p-value histograms and volcano plots (fold change vs. p-value).

Usage

```
pdistr(eset, out.file=NULL, use.adjp=TRUE)
volcano(eset, out.file=NULL)
exprs.heatmap(eset, out.file=NULL)
```

Arguments

`eset`	Expression set. An object of class 'ExpressionSet' containing the gene expression set. See 'read.eset' and 'probe.2.gene.eset' for required annotations in the pData and fData slot.
`use.adjp`	Logical. Use differential expression p-values that are already corrected for multiple testing? Defaults to TRUE.
`out.file`	Optional output file to which graphics are plotted to.

Value

None, plots to a graphics device.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

See Also

`heatmap`, `truehist`

Examples

```
# (1) reading the expression data from file
exprs.file <- system.file("extdata/ALL_exprs.tab", package="EnrichmentBrowser")
pdat.file <- system.file("extdata/ALL_pData.tab", package="EnrichmentBrowser")
fdat.file <- system.file("extdata/ALL_fData.tab", package="EnrichmentBrowser")
eset <- read.eset(exprs.file, pdat.file, fdat.file)

# (2) plot
exprs.heatmap(eset)
pdistr(eset)
volcano(eset)
```

`probe.2.gene.eset` *Transformation of probe level expression to gene level expression*

Description

Reads expression data at probe level and summarizes gene expression behavior by averaging over all probes that are annotated to a particular gene. A subsequent differential expression analysis at the gene level is automatically performed.

Usage

```
probe.2.gene.eset(probe.eset, use.mean=TRUE,
    heatm.file=NULL, distr.file=NULL, volc.file=NULL)
```

Arguments

`probe.eset`	Probe expression set of class 'ExpressionSet'. The fData slot of the expression set must contain a 'GENE' column that lists for each probe the corresponding KEGG gene ID.
`use.mean`	Logical. Determining, in case of multiple probes for one gene, whether a mean value is computed or the probe that discriminate the most between the two sample group is kept. Defaults to TRUE.
`heatm.file`	Optional. If specified, a heatmap is plotted in PNG format to file.
`distr.file`	Optional. If specified, the p-value distribution is plotted in PNG format to file.
`volc.file`	Optional. If specified, the volcano plot is plotted in PNG format to file.

Value

An `ExpressionSet-class` on gene level with measures of differential expression annotated.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

See Also

`ExpressionSet-class`, `eBayes`, `p.adjust`

Examples

```
# (1) reading the expression data from file
exprs.file <- system.file("extdata/ALL_exprs.tab", package="EnrichmentBrowser")
pdat.file <- system.file("extdata/ALL_pData.tab", package="EnrichmentBrowser")
fdat.file <- system.file("extdata/ALL_fData.tab", package="EnrichmentBrowser")
probe.eset <- read.eset(exprs.file, pdat.file, fdat.file)
gene.eset <- probe.2.gene.eset(probe.eset)
head(fData(gene.eset))
```

`read.eset` *Reading gene expression data from file into an expression set*

Description

The function reads in plain expression data with minimum annotation requirements for the pData and fData slots. A differential expression analysis can be performed and annotated to the expression set. If desired, overview plots such as a heatmap, p-value distribution and volcano plot (fold change vs. p-value) are created.

Usage

```
read.eset(exprs.file, pdat.file, fdat.file, de=TRUE,
            heatm.file=NULL, distr.file = NULL, volc.file = NULL)
```

Arguments

`exprs.file`	Expression matrix. A tab separated text file containing *normalized* expression values on a *log* scale. Columns = samples/subjects; rows = features/probes/genes; NO headers, row or column names. Supported data types are log2 counts (microarray single-channel), log2 ratios (microarray two-color), and log2-counts per million (RNA-seq logCPMs). See details.
`pdat.file`	Phenotype data. A tab separated text file containing annotation information for the samples in either *two or three* columns. NO headers, row or column names. The number of rows/samples in this file should match the number of columns/samples of the expression matrix. The 1st colum is reserved for the sample IDs; The 2nd column is reserved for a *BINARY* group assignment. Use '0' and '1' for unaffected (controls) and affected (cases) sample class, respectively. For paired samples or sample blocks a third column is expected that defines the blocks.
`fdat.file`	Feature data. A tab separated text file containing annotation information for the features. Exactly *TWO* columns; 1st col = feature IDs; 2nd col = corresponding KEGG gene ID for each feature ID in 1st col; NO headers, row or column names. The number of rows/features in this file should match the number of rows/features of the expression matrix. Alternatively, this can also be the ID of a recognized platform such as 'hgu95av2' (Affymetrix Human Genome U95 chip) or 'ecoli2' (Affymetrix E. coli Genome 2.0 Array). See details.
`de`	Logical. Should a simultanous differential expression (de) analysis be carried out for each gene. Defaults to TRUE.
`heatm.file`	Optional. If specified, a heatmap is plotted in PNG format to file.
`distr.file`	Optional. If specified, the p-value distribution is plotted in PNG format to file.
`volc.file`	Optional. If specified, the volcano is plotted in PNG format to file.

Details

See the limma's user guide http://www.bioconductor.org/packages/release/bioc/vignettes/limma/inst/doc/usersguide.pdf for definition and normalization of the different expression data types.

In case of microarry data the feature IDs typically correspond to probe IDs. Thus, the fdat.file should define a mapping from probe ID (1st column) to corresponding KEGG gene ID (2nd column). The mapping can be defined automatically by providing the ID of a recognized platform such as 'hgu95av2' (Affymetrix Human Genome U95 chip). This requires that a corresponding '.db' package exists (see http://www.bioconductor.org/packages/release/BiocViews.html#___ChipName for all available chips/packages) and that you have it installed. *However, this option should be used with care*. Existing mappings might be outdated and sometimes the KEGG gene ID does not correspond to the Entrez ID (e.g. for E. coli and S. cerevisae). In these cases probe identifiers are mapped twice (probe ID -> Entrez ID -> KEGG ID) which almost always results in loss of information. Thus, mapping quality should always be checked and in case properly defined with a 2 column fdat.file.

Value

An ExpressionSet-class with measures of differential expression annotated in the fData slot.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

See Also

ExpressionSet-class, eBayes, voom, p.adjust

Examples

```
# reading the expression data from file
exprs.file <- system.file("extdata/ALL_exprs.tab", package="EnrichmentBrowser")
pdat.file <- system.file("extdata/ALL_pData.tab", package="EnrichmentBrowser")
fdat.file <- system.file("extdata/ALL_fData.tab", package="EnrichmentBrowser")
eset <- read.eset(exprs.file, pdat.file, fdat.file)
head(fData(eset))
```

---

sbea *Set-based enrichment analysis (SBEA)*

Description

This is the main function for the enrichment analysis of gene sets. It implements and uses existing implementations of several frequently used state-of-art methods and allows a flexible inspection of resulting gene set rankings.

Usage

```
sbea(method=c("ora", "safe", "gsea", "samgs"), eset, gs,
        alpha=0.05, perm=1000, out.file=NULL, browse=FALSE)

sbea.methods()
```

Arguments

method	Set-based enrichment analysis method. Currently, the following set-based enrichment analysis methods are supported: ‘ora’, ‘safe’, ‘gsea’, and ‘samgs’. See Details. For basic ora also set ’perm=0’. Default is ‘ora’. This can also be the name of a tailored function implementing set-based enrichment. See Details.
eset	Expression set. Either an object of class ’ExpressionSet’ or an absolute file path to an RData file containing the gene expression set. See ’read.eset’ and ’probe.2.gene.eset’ for required annotations in the pData and fData slot.
gs	Gene sets. Either a list of gene sets (vectors of KEGG gene IDs) or a text file in GMT format storing all gene sets under investigation.

`alpha`	Statistical significance level. Defaults to 0.05.
`perm`	Number of permutations of the expression matrix to estimate the null distribution. Defaults to 1000. For basic ora set 'perm=0'.
`out.file`	Optional output file the gene set ranking will be written to.
`browse`	Logical. Should results be displayed in the browser for interactive exploration? Defaults to FALSE.

Details

'ora': overrepresentation analysis, simple and frequently used test based on the hypergeometric distribution (see Goeman and Buhlmann, 2007, for a critical review). 'safe': significance analysis of function and expression, generalization of ORA, includes other test statistics, e.g. Wilcoxon's rank sum, and allows to estimate the significance of gene sets by sample permutation; implemented in the safe package (Barry et al., 2005). 'gsea': gene set enrichment analysis, frequently used and widely accepted, uses a Kolmogorov-Smirnov statistic to test whether the ranks of the p-values of genes in a gene set resemble a uniform distribution (Subramanian et al., 2005). 'samgs': significance analysis of microarrays on gene sets, extends the SAM method for single genes to gene set analysis (Dinu et al., 2007).

It is also possible to use additional set-based enrichment methods. This requires to implement a function that takes 'eset', 'gs', 'alpha', and 'perm' as arguments and returns a numeric vector 'ps' storing the resulting p-value for each gene set in 'gs'. This vector must be named accordingly (i.e. names(ps) == names(gs)). See examples.

Value

sbea.methods: a character vector of currently supported methods;

sbea: if(is.null(out.file)): an enrichment analysis result object that can be detailedly explored by calling 'ea.browse' and from which a flat gene set ranking can be extracted by calling 'gs.ranking'. If 'out.file' is given, the ranking is written to the specified file.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@bio.ifi.lmu.de>

References

Goeman and Buhlmann (2007) Analyzing gene expression data in terms of gene sets: methodological issues. Bioinformatics, 23, 980-7.

Barry et al. (2005) Significance Analysis of Function and Expression. Bioinformatics, 21:1943-9.

Subramanian et al. (2005) Gene Set Enrichment Analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proc Natl Acad Sci USA, 102:15545-50.

Dinu et al. (2007) Improving gene set analysis of microarray data by SAM-GS. BMC Bioinformatics, 8:242

See Also

Input: `read.eset`, `probe.2.gene.eset` `get.kegg.genesets` to retrieve gene sets from KEGG.

Output: `gs.ranking` to retrieve the ranked list of gene sets. `ea.browse` for exploration of resulting gene sets.

Other: `nbea` to perform network-based enrichment analysis. `comb.ea.results` to combine results from different methods.

Examples

```
# currently supported methods
sbea.methods()

# (1) reading the expression data from file
exprs.file <- system.file("extdata/ALL_exprs.tab", package="EnrichmentBrowser")
pdat.file <- system.file("extdata/ALL_pData.tab", package="EnrichmentBrowser")
fdat.file <- system.file("extdata/ALL_fData.tab", package="EnrichmentBrowser")
probe.eset <- read.eset(exprs.file, pdat.file, fdat.file)

# (2) summarizing probe expression on gene level
gene.eset <- probe.2.gene.eset(probe.eset)

# (3) getting all human KEGG gene sets
# hsa.gs <- get.kegg.genesets("hsa")
gs.file <- system.file("extdata/hsa_kegg_gs.gmt", package="EnrichmentBrowser")
hsa.gs <- parse.genesets.from.GMT(gs.file)

# (4) performing the enrichment analysis
ea.res <- sbea(method="ora", eset=gene.eset, gs=hsa.gs, perm=0)

# (5) result visualization and exploration
gs.ranking(ea.res)

ea.browse(ea.res)


# using your own tailored function as enrichment method
dummy.sbea <- function(eset, gs, alpha, perm)
{
    sig.ps <- sample(seq(0,0.05, length=1000),5)
    insig.ps <- sample(seq(0.1,1, length=1000), length(gs)-5)
    ps <- sample(c(sig.ps, insig.ps), length(gs))
    names(ps) <- names(gs)
    return(ps)
}

sbea.res2 <- sbea(method="dummy.sbea", eset=gene.eset, gs=hsa.gs)
gs.ranking(sbea.res2)
```

Index