Package ‘GSEABenchmarkeR’

October 16, 2019

Type Package

Title Reproducible GSEA Benchmarking

Version 1.4.0

Author Ludwig Geistlinger [aut, cre],
Gergely Csaba [aut],
Mara Santarelli [ctb],
Lucas Schiffer [ctb],
Marcel Ramos [ctb],
Ralf Zimmer [aut],
Levi Waldron [aut]

Maintainer Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

Description The GSEABenchmarkeR package implements an extendable framework for reproducible evaluation of set- and network-based methods for enrichment analysis of gene expression data. This includes support for the efficient execution of these methods on comprehensive real data compendia (microarray and RNA-seq) using parallel computation on standard workstations and institutional computer grids. Methods can then be assessed with respect to runtime, statistical significance, and relevance of the results for the phenotypes investigated.

URL https://github.com/waldronlab/GSEABenchmarkeR

BugReports https://github.com/waldronlab/GSEABenchmarkeR/issues

License Artistic-2.0

Encoding UTF-8

LazyData true

Depends Biobase, SummarizedExperiment

Imports AnnotationDbi, AnnotationHub, BiocFileCache, BiocParallel, edgeR, EnrichmentBrowser, ExperimentHub, GEOquery, grDevices, graphics, KEGGandMetacoreDzPathwaysGEO, KEGGdzPathwaysGEO, methods, rappdirs, S4Vectors, stats, utils

Suggests BiocStyle, knitr, rmarkdown

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

VignetteBuilder knitr

RoxygenNote 6.1.0

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

git_branch RELEASE_3_9

git_last_commit 11383ab

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

R topics documented:

`bpPlot` *Customized boxplot visualization of benchmark results*

Description

This is a convenience function to create customized boxplots for specific benchmark criteria such as runtime, statistical significance and phenotype relevance.

Usage

```
bpPlot(data, what = c("runtime", "sig.sets", "rel.sets", "typeI"))
```

Arguments

`data`	Numeric matrix or list of numeric vectors. In case of a matrix, column names are assumed to be method names and rownames are assumed to be dataset IDs. In case of a list, names are assumed to be methods names and each element corresponds to a numeric vector with names assumed to be dataset IDs.
`what`	Character. Determines how the plot is customized. One of • runtime: displays runtime of methods across datasets, • sig.sets: displays percentage of significant gene sets, • rel.sets: displays phenotype relevance scores.

Value

None. Plots to a graphics device.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

`evalNrSigSets` to evaluate fractions of significant gene sets; `evalRelevance` to evaluate phenotype relevance of gene set rankings.

Examples

```
# simulated setup:
# 3 methods & 5 datasets
methods <- paste0("m", 1:3)
data.ids <- paste0("d", 1:5)

# runtime data
rt <- vapply(1:3, function(m) runif(5, min=m, max=m+1), numeric(5))
rownames(rt) <- data.ids
colnames(rt) <- methods

# plot
bpPlot(rt, what="runtime")
```

`cacheResource` *Caching of a resource*

Description

Convenience function to flexibly save and restore an already processed expression data compendium via caching.

Usage

```
cacheResource(res, rname, ucdir = "GSEABenchmarkeR")
```

Arguments

`res`	Resource. An arbitrary R object.
`rname`	Character. Resource name.
`ucdir`	Character. User cache directory. Defaults to 'GSEABenchmarkeR', which will accordingly use `rappdirs::user_cache_dir("GSEABenchmarkeR")`.

Value

None. Stores the object in the cache by side effect.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

loadEData, user_cache_dir, BiocFileCache

Examples

```
# load user-defined expression compendium
data.dir <- system.file("extdata/myEData", package="GSEABenchmarkeR")
edat <- loadEData(data.dir)

# do some processing of the compendium
edat <- lapply(edat, function(d) d[1:50,])

# cache it ...
cacheResource(edat, "myEData")

# ... and restore it at a later time
edat <- loadEData(data.dir, cache=TRUE)
```

`evalNrSigSets` *Evaluating gene set rankings for the number of (significant) sets*

Description

These functions evaluate gene set rankings obtained from applying enrichment methods to multiple datasets. This allows to assess resulting rankings for granularity (how many gene sets have a unique p-value?) and statistical significance (how many gene sets have a p-value below a significance threshold?).

Usage

```
evalNrSigSets(ea.ranks, alpha = 0.05, padj = "none", perc = TRUE)

evalNrSets(ea.ranks, uniq.pval = TRUE, perc = TRUE)
```

Arguments

ea.ranks	Enrichment analysis rankings. A list with an entry for each enrichment method applied. Each entry is a list that stores for each dataset analyzed the resulting gene set ranking as obtained from applying the respective method to the respective dataset.
alpha	Statistical significance level. Defaults to 0.05.
padj	Character. Method for adjusting p-values to multiple testing. For available methods see the man page of the stats function p.adjust. Defaults to BH.
perc	Logical. Should the percentage or absolute number of gene sets be returned? Percentage is typically more useful for comparison between rankings with a potentially different total number of gene sets. Defaults to TRUE.
uniq.pval	Logical. Should the number of gene sets with a unique p-value or the total number of gene sets per ranking be returned? Defaults to TRUE.

Value

A list of numeric vectors storing for each method the number of (significant) gene sets for each dataset analyzed. If each element of the resulting list is of equal length (corresponds to successful application of each enrichment method to each dataset), the list is automatically simplified to a numeric matrix (rows = datasets, columns = methods).

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

runEA to apply enrichment methods to multiple datasets; readResults to read saved rankings as an input for the eval-functions.

Examples

```
# simulated setup:
# 2 methods & 2 datasets
methods <- paste0("m", 1:2)
data.ids <- paste0("d", 1:2)

# simulate gene set rankings
ea.ranks <- sapply(methods, function(m)
        sapply(data.ids,
            function(d)
            {
                r <- EnrichmentBrowser::makeExampleData("ea.res")
                r <- EnrichmentBrowser::gsRanking(r, signif.only=FALSE)
                return(r)
            }, simplify=FALSE),
            simplify=FALSE)

# evaluate
evalNrSets(ea.ranks)
evalNrSigSets(ea.ranks)
```

---

evalRelevance *Evaluating phenotype relevance of gene set rankings*

Description

This function evaluates gene set rankings obtained from the application of enrichment methods to multiple datasets - where each dataset investigates a certain phenotype such as a disease. Given pre-defined phenotype relevance scores for the gene sets, indicating how important a gene set is for the investigated phenotype (as e.g. judged by evidence from the literature), this allows to assess whether enrichment methods produce gene set rankings in which phenotype-relevant gene sets accumulate at the top.

Usage

```
evalRelevance(ea.ranks, rel.ranks, data2pheno, perc = TRUE, top = 0,
  rand = FALSE)

compOpt(rel.ranks, gs.ids, data2pheno = NULL, top = 0)

compRand(rel.ranks, gs.ids, data2pheno = NULL, perm = 1000)
```

Arguments

`ea.ranks`	Enrichment analysis rankings. A list with an entry for each enrichment method applied. Each entry is a list that stores for each dataset analyzed the resulting gene set ranking obtained from applying the respective method to the respective dataset. Resulting gene set rankings are assumed to be of class `DataFrame` in which gene sets (required column named `GENE.SET`) are ranked according to a ranking measure such as a gene set p-value (required column named `P.VALUE`). See `gsRanking` for an example.
`rel.ranks`	Relevance score rankings. A list with an entry for each phenotype investigated. Each entry should be a `DataFrame` in which gene sets (rownames are assumed to be gene set IDs) are ranked according to a phenotype relevance score (required column `REL.SCORE`).
`data2pheno`	A named character vector where the names correspond to dataset IDs and the elements of the vector to the corresponding phenotypes investigated.
`perc`	Logical. Should observed scores be returned as-is or as a *perc*entage of the respective optimal score. Percentages of the optimal score are typically easier to interpret and are comparable between datasets / phenotypes. Defaults to `TRUE`.
`top`	Integer. If `top` is non-zero, the evaluation will be restricted to the first `top` gene sets of each enrichment analysis ranking. Defaults to `0`, which will then evaluate the full ranking.
`rand`	Logical. Should gene set rankings be randomized to assess how likely it is to observe a score equal or greater than the respective obtained score? Defaults to `FALSE`.
`gs.ids`	Character vector of gene set IDs on which enrichment analysis has been carried out.
`perm`	Integer. Number of permutations if `rand` set to `TRUE`.

Details

The function `evalRelevance` evaluates the similarity of a gene set ranking obtained from enrichment analysis and a gene set ranking based on phenotype relevance scores. Therefore, the function first transforms the ranks 'r' from the enrichment analysis to weights 'w' in [0,1] via w = 1 - r / N; where 'N' denotes the total number of gene sets on which the enrichment analysis has been carried out. These weights are then multiplied with the corresponding relevance scores and summed up.

The function `compOpt` applies `evalRelevance` to the theoretically optimal case in which the enrichment analysis ranking is identical to the relevance score ranking. The ratio between observed and optimal score is useful for comparing observed scores between datasets / phenotypes.

The function `compRand` repeatedly applies `evalRelevance` to randomly drawn gene set rankings to assess how likely it is to observe a score equal or greater than the one obtained.

Value

A numeric matrix (rows = datasets, columns = methods) storing in each cell the relevance score sum obtained from applying the respective method to the respective dataset.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

`runEA` to apply enrichment methods to multiple datasets; `readResults` to read saved rankings as an input for the eval-functions;

Examples

```
#
# (1) simulated setup: 1 enrichment method applied to 1 dataset
#

# simulate gene set ranking
ea.ranks <- EnrichmentBrowser::makeExampleData("ea.res")
ea.ranks <- EnrichmentBrowser::gsRanking(ea.ranks, signif.only=FALSE)

# simulated relevance score ranking
rel.ranks <- ea.ranks
rel.ranks[,2] <- runif(nrow(ea.ranks), min=1, max=100)
colnames(rel.ranks)[2] <- "REL.SCORE"
rownames(rel.ranks) <- rel.ranks[,"GENE.SET"]
ind <- order(rel.ranks[,"REL.SCORE"], decreasing=TRUE)
rel.ranks <- rel.ranks[ind,]

# evaluate
evalRelevance(ea.ranks, rel.ranks)
compOpt(rel.ranks, ea.ranks[,"GENE.SET"])
compRand(rel.ranks, ea.ranks[,"GENE.SET"], perm=3)

#
# (2) simulated setup: 2 methods & 2 datasets
#

methods <- paste0("m", 1:2)
data.ids <- paste0("d", 1:2)

# simulate gene set rankings
ea.ranks <- sapply(methods, function(m)
        sapply(data.ids,
            function(d)
            {
                r <- EnrichmentBrowser::makeExampleData("ea.res")
                r <- EnrichmentBrowser::gsRanking(r, signif.only=FALSE)
                return(r)
            }, simplify=FALSE),
            simplify=FALSE)

# simulate a mapping from datasets to disease codes
```

```
d2d <- c("ALZ", "BRCA")
names(d2d) <- data.ids

# simulate relevance score rankings
rel.ranks <- lapply(ea.ranks[[1]],
    function(rr)
    {
        rr[,2] <- runif(nrow(rr), min=1, max=100)
        colnames(rr)[2] <- "REL.SCORE"
        rownames(rr) <- rr[,"GENE.SET"]
        ind <- order(rr[,"REL.SCORE"], decreasing=TRUE)
        rr <- rr[ind,]
        return(rr)
    })
names(rel.ranks) <- unname(d2d)

# evaluate
evalRelevance(ea.ranks, rel.ranks, d2d)
```

`evalTypeIError` *Evaluation of the type I error rate of enrichment methods*

Description

This function evaluates the type I error rate of selected methods for enrichment analysis when applied to one or more expression datasets.

Usage

```
evalTypeIError(methods, exp.list, gs, alpha = 0.05, ea.perm = 1000,
  tI.perm = 1000, perm.block.size = -1, parallel = NULL,
  save2file = FALSE, out.dir = NULL, ...)
```

Arguments

`methods`	Methods for enrichment analysis. A character vector with method names chosen from `sbeaMethods` and `nbeaMethods`, or user-defined functions implementing methods for enrichment analysis.
`exp.list`	Experiment list. A `list` of datasets, each being of class `SummarizedExperiment`.
`gs`	Gene sets, i.e. a list of character vectors of gene IDs.
`alpha`	Numeric. Statistical significance level. Defaults to 0.05.
`ea.perm`	Integer. Number of permutations of the sample group assignments during enrichment analysis. Defaults to 1000. Can also be an integer vector matching the length of 'methods' to assign different numbers of permutations for different methods.
`tI.perm`	Integer. Number of permutations of the sample group assignments during type I error rate evaluation. Defaults to 1000. Can also be an integer vector matching the length of `methods` to assign different numbers of permutations for different methods.

perm.block.size
: Integer. When running in parallel, splits `tI.perm` into blocks of the indicated size. Defaults to -1, which indicates to not partition `tI.perm`.

parallel
: Parallel computation mode. An instance of class `BiocParallelParam`. See the vignette of the `BiocParallel` package for switching between serial, multi-core, and grid execution. Defaults to `NULL`, which then uses the first element of `BiocParallel::registered()` for execution. If not changed by the user, this accordingly defaults to multi-core execution on the local host.

save2file
: Logical. Should results be saved to file for subsequent benchmarking? Defaults to `FALSE`.

out.dir
: Character. Determines the output directory where results are saved to. Defaults to `NULL`, which then writes to `rappdirs::user_data_dir("GSEABenchmarkeR")` in case `save2file` is set to `TRUE`.

...
: Additional arguments passed to the selected enrichment methods.

Value

A list with an entry for each method applied. Each method entry is a list with an entry for each dataset analyzed. Each dataset entry is a list of length 2, with the first element being the runtime and the second element being the gene set ranking, as obtained from applying the respective method to the respective dataset.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

`sbea` and `nbea` for carrying out set- and network-based enrichment analysis.

`BiocParallelParam` and `register` for configuration of parallel computation.

Examples

```
# loading three datasets from the GEO2KEGG compendium
geo2kegg <- loadEData("geo2kegg", nr.datasets=3)

# only considering the first 1000 probes for demonstration
geo2kegg <- lapply(geo2kegg, function(d) d[1:1000,])

# preprocessing and DE analysis for two of the datasets
geo2kegg <- maPreproc(geo2kegg[2:3])
geo2kegg <- runDE(geo2kegg)

# getting a subset of human KEGG gene sets
gs.file <- system.file("extdata", package="EnrichmentBrowser")
gs.file <- file.path(gs.file, "hsa_kegg_gs.gmt")
kegg.gs <- EnrichmentBrowser::getGenesets(gs.file)

# evaluating type I error rate of two methods on two datasets
# NOTE: using a small number of permutations for demonstration;
#       for a meaningful evaluation tI.perm should be >= 1000
res <- evalTypeIError(geo2kegg, methods=c("ora",
         "camera"), gs=kegg.gs, ea.perm=0, tI.perm=3)
```

`loadEData` *Loading pre-defined and user-defined expression data*

Description

This function implements a general interface for loading the pre-defined GEO2KEGG microarray compendium and the TCGA RNA-seq compendium. It also allows loading of user-defined data from file.

Usage

```
loadEData(edata, nr.datasets = NULL, cache = TRUE, ...)
```

Arguments

`edata` Expression data compendium. A character vector of length 1 that must be either

- 'geo2kegg': to load the GEO2KEGG microarray compendium,
- 'tcga': to load the TCGA RNA-seq compendium, or
- an absolute file path pointing to a directory, in which a user-defined compendium has been saved in RDS files.

See details.

`nr.datasets` Integer. Number of datasets that should be loaded from the compendium. This is mainly for demonstration purposes.

`cache` Logical. Should an already cached version used if available? Defaults to `TRUE`.

`...` Additional arguments passed to the internal loading routines of the GEO2KEGG and TCGA compendia. This currently includes for loading of the GEO2KEGG compendium

- preproc: logical. Should probe level data automatically be summarized to gene level data? Defaults to FALSE.
- de.only: logical. Include only datasets in which differentially expressed genes have been found? Defaults to FALSE.
- excl.metac: logical. Exclude datasets for which MetaCore rather than KEGG pathways have been assigned as target pathways? Defaults to FALSE.

And for loading of the TCGA compendium

- mode: character, determines how GSE62944 is obtained. Either 'ehub' (default, via ExperimentHub) or 'geo' (direct download from GEO, slow).
- data.dir: character. Absolute file path indicating where processed RDS files for each dataset are written to. Defaults to NULL, which will then write to `rappdirs::user_data_dir("GSEABenchmarkeR")`.
- min.ctrls: integer. Minimum number of controls, i.e. adjacent normal samples, for a cancer type to be included. Defaults to 9.
- min.cpm: integer. Minimum counts-per-million reads mapped. See the edgeR vignette for details. The default filter is to exclude genes with cpm < 2 in more than half of the samples.
- with.clin.vars: logical. Should clinical variables (>500) be kept to allow for more advanced sample groupings in addition to the default binary grouping (tumor vs. normal)?

Details

The pre-defined GEO2KEGG microarray compendium consists of 42 datasets investigating a total of 19 different human diseases as collected by Tarca et al. (2012 and 2013).

The pre-defined TCGA RNA-seq compendium consists of datasets from The Cancer Genome Atlas (TCGA, 2013) investigating a total of 34 different cancer types.

User-defined data can also be loaded, given that datasets, preferably of class `SummarizedExperiment`, have been saved as `RDS` files.

Value

A `list` of datasets, typically of class `SummarizedExperiment`.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

References

Tarca et al. (2012) Down-weighting overlapping genes improves gene set analysis. BMC Bioinformatics, 13:136.

Tarca et al. (2013) A comparison of gene set analysis methods in terms of sensitivity, prioritization and specificity. PLoS One, 8(11):e79217.

The Cancer Genome Atlas Research Network (2013) The Cancer Genome Atlas Pan-Cancer analysis project. Nat Genet, 45(10):1113-20.

Rahman et al. (2015) Alternative preprocessing of RNA-Sequencing data in The Cancer Genome Atlas leads to improved analysis results. Bioinformatics, 31(22):3666-72.

See Also

`SummarizedExperiment`

Examples

```
# (1) Loading the GEO2KEGG microarray compendium
geo2kegg <- loadEData("geo2kegg", nr.datasets=2)

# (2) Loading the TCGA RNA-seq compendium
tcga <- loadEData("tcga", nr.datasets=2)

# (3) reading user-defined expression data from file
data.dir <- system.file("extdata/myEData", package="GSEABenchmarkeR")
edat <- loadEData(data.dir)
```

`maPreproc` *Preprocessing of microarray expression data*

Description

This function prepares datasets of the GEO2KEGG microarray compendium for further analysis. This includes summarization of probe level expression to gene level expression as well as annotation of required colData slots for sample grouping.

Usage

```
maPreproc(exp.list, parallel = NULL)
```

Arguments

`exp.list`	Experiment list. A `list` of datasets, each being of class `ExpressionSet`.
`parallel`	Parallel computation mode. An instance of class `BiocParallelParam`. See the vignette of the `BiocParallel` package for switching between serial, multi-core, and grid execution. Defaults to `NULL`, which then uses the first element of `BiocParallel::registered()` for execution. If not changed by the user, this accordingly defaults to multi-core execution on the local host.

Value

A `list` of datasets, each being of class `SummarizedExperiment`.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

`loadEData` to load a specified expression data compendium.

Examples

```
# reading user-defined expression data from file
geo2kegg <- loadEData("geo2kegg", nr.datasets=3)

# only considering the first 100 probes for demonstration
geo2kegg <- lapply(geo2kegg, function(d) d[1:100,])

# preprocessing two datasets
geo2kegg <- maPreproc(geo2kegg[2:3])
```

`readDataId2diseaseCodeMap` *Read a mapping between dataset ID and disease code*

Description

When assessing enrichment analysis results for phenotype relevance, it is assumed that each analyzed dataset investigates a certain phenotype such as a disease. This function reads a mapping between dataset IDs and assigned disease codes.

Usage

```
readDataId2diseaseCodeMap(map.file)
```

Arguments

`map.file`	Character. The path to the mapping file.

Value

A named character vector where each element of the vector is a disease code and the names are the dataset IDs.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

`evalApply` for evaluating phenotype relevance of gene set rankings.

Examples

```
data.dir <- system.file("extdata", package="GSEABenchmarkeR")
d2d.file <- file.path(data.dir, "malacards", "GseId2Disease.txt")
d2d.map <- readDataId2diseaseCodeMap(d2d.file)
```

`readResults` *Reading results of enrichment analysis*

Description

These functions read results obtained from the application of enrichment methods to multiple datasets for subsequent assessment.

Usage

```
readResults(data.dir, data.ids, methods, type = c("runtime", "ranking",
  "typeI"))
```

Arguments

`data.dir`	Character. The data directory where results have been saved to.
`data.ids`	A character vector of dataset IDs.
`methods`	Methods for enrichment analysis. A character vector with method names typically chosen from `sbeaMethods` and `nbeaMethods`, or user-defined functions implementing methods for enrichment analysis.
`type`	Character. Type of the result. Should be one out of 'runtime', 'ranking', or 'typeI'.

Value

A result list with an entry for each method applied. Each entry stores corresponding runtimes (`type="runtime"`), gene set rankings (`type="ranking"`), or type I error rates (`type="typeI"`) as obtained from applying the respective method to the given datasets.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

`runEA` to apply enrichment methods to multiple datasets.

Examples

```
    # simulated setup:
    # 1 methods & 1 datasets
    methods <- paste0("m", 1:2)
    data.ids <- paste0("d", 1:2)

    # result directory
    res.dir <- tempdir()
    sdirs <- file.path(res.dir, methods)
    for(d in sdirs) dir.create(d)

    # store runtime & rankings
    for(m in 1:2)
    {
        rt <- runif(5, min=m, max=m+1)
        for(d in 1:2)
        {
            # runtime
            out.file <- paste(data.ids[d], "txt", sep=".")
            out.file <- file.path(sdirs[m], out.file)
            cat(rt[d], file=out.file)

            # ranking
            out.file <- sub("txt$", "rds", out.file)
            r <- EnrichmentBrowser::makeExampleData("ea.res")
            r <- EnrichmentBrowser::gsRanking(r, signif.only=FALSE)
            saveRDS(r, file=out.file)
        }
    }
```

```
# reading runtime & rankings
rts <- readResults(res.dir, data.ids, methods, type="runtime")
rkgs <- readResults(res.dir, data.ids, methods, type="ranking")
```

---

runDE *Differential expression analysis for datasets of a compendium*

Description

This function applies selected methods for differential expression (DE) analysis to selected datasets of an expression data compendium.

Usage

```
runDE(exp.list, de.method = c("limma", "edgeR", "DESeq2"),
  padj.method = "flexible", parallel = NULL, ...)

writeDE(exp.list, out.dir = NULL)

plotDEDistribution(exp.list, alpha = 0.05, beta = 1)
```

Arguments

`exp.list`	Experiment list. A `list` of datasets, each being of class `SummarizedExperiment`.
`de.method`	Differential expression method. See documentation of `deAna`.
`padj.method`	Method for adjusting p-values to multiple testing. For available methods see the man page of the stats function `p.adjust`. Defaults to 'flexible', which applies a dataset-specific correction strategy. See details.
`parallel`	Parallel computation mode. An instance of class `BiocParallelParam`. See the vignette of the `BiocParallel` package for switching between serial, multi-core, and grid execution. Defaults to `NULL`, which then uses the first element of `BiocParallel::registered()` for execution. If not changed by the user, this accordingly defaults to multi-core execution on the local host.
`...`	Additional arguments passed to `EnrichmentBrowser::deAna`.
`out.dir`	Character. Determines the output directory where DE results for each dataset are written to. Defaults to `NULL`, which then writes to a subdir named 'de' in `rappdirs::user_data_dir("GSEABenchmarkeR")`.
`alpha`	Statistical significance level. Defaults to 0.05.
`beta`	Absolute log2 fold change cut-off. Defaults to 1 (2-fold).

Details

DE studies typically report a gene as differentially expressed if the corresponding DE p-value, corrected for multiple testing, satisfies the chosen significance level. Enrichment methods that work directly on the list of DE genes are then substantially influenced by the multiple testing correction.

An example is the frequently used over-representation analysis (ORA), which assesses the overlap between the DE genes and a gene set under study based on the hypergeometric distribution (see Appendix A of the `EnrichmentBrowser` vignette for an introduction).

ORA is inapplicable if there are few genes satisfying the significance threshold, or if almost all genes are DE.

Using `padj.method="flexible"` accounts for these cases by applying multiple testing correction in dependence on the degree of differential expression:

- the correction method from Benjamini and Hochberg (BH) is applied if it renders >= 1% and <= 25% of all measured genes as DE,
- the p-values are left unadjusted, if the BH correction results in < 1% DE genes, and
- the more stringent Bonferroni correction is applied, if the BH correction results in > 25% DE genes.

Note that resulting p-values should not be used for assessing the statistical significance of DE genes within or between datasets. They are solely used to determine which genes are included in the analysis with ORA - where the flexible correction ensures that the fraction of included genes is roughly in the same order of magnitude across datasets.

Alternative stratgies could also be applied - such as taking a constant number of genes for each dataset or excluding ORA methods in general from the assessment.

Value

`runDE` returns `exp.list` with DE measures annotated to the `rowData` slot of each dataset, `writeDE` writes to file, and `plotDEDistribution` plots to a graphics device.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

`loadEData` to load a specified expression data compendium.

Examples

```
# reading user-defined expression data from file
data.dir <- system.file("extdata/myEData", package="GSEABenchmarkeR")
edat <- loadEData(data.dir)

# differential expression analysis
edat <- runDE(edat)

# visualization of per-dataset DE distribution
plotDEDistribution(edat)

# writing DE results to file
out.dir <- tempdir()
out.dir <- file.path(out.dir, "de")
if(!file.exists(out.dir)) dir.create(out.dir)

writeDE(edat, out.dir)
```

runEA *Application of enrichment methods to multiple datasets*

Description

This function applies selected methods for enrichment analysis to selected datasets of a compendium.

Usage

```
runEA(exp.list, methods, gs, perm = 1000, parallel = NULL,
  save2file = FALSE, out.dir = NULL, ...)
```

Arguments

`exp.list`	Experiment list. A `list` of datasets, each being of class `SummarizedExperiment`. In case of just one dataset a single `SummarizedExperiment` is also allowed. See the documentation of `sbea` for required minimal annotations.
`methods`	Methods for enrichment analysis. A character vector with method names chosen from `sbeaMethods` and `nbeaMethods`, or user-defined functions implementing methods for enrichment analysis.
`gs`	Gene sets, i.e. a list of character vectors of gene IDs.
`perm`	Number of permutations of the sample group assignments. Defaults to 1000. Can also be an integer vector matching the length of `methods` to assign different numbers of permutations for different methods.
`parallel`	Parallel computation mode. An instance of class `BiocParallelParam`. See the vignette of the `BiocParallel` package for switching between serial, multi-core, and grid execution. Defaults to `NULL`, which then uses the first element of `BiocParallel::registered()` for execution. If not changed by the user, this accordingly defaults to multi-core execution on the local host.
`save2file`	Logical. Should results be saved to file for subsequent benchmarking? Defaults to `FALSE`.
`out.dir`	Character. Determines the output directory where results are saved to. Defaults to `NULL`, which then writes to `rappdirs::user_data_dir("GSEABenchmarkeR")` in case `save2file` is set to `TRUE`.
`...`	Additional arguments passed to the selected enrichment methods.

Value

A list with an entry for each method applied. Each method entry is a list with an entry for each dataset analyzed. Each dataset entry is a list of length 2, with the first element being the runtime and the second element being the gene set ranking, as obtained from applying the respective method to the respective dataset.

Author(s)

Ludwig Geistlinger <Ludwig.Geistlinger@sph.cuny.edu>

See Also

sbea and nbea for carrying out set- and network-based enrichment analysis.

BiocParallelParam and register for configuration of parallel computation.

Examples

```
# loading three datasets from the GEO2KEGG compendium
geo2kegg <- loadEData("geo2kegg", nr.datasets=3)

# only considering the first 1000 probes for demonstration
geo2kegg <- lapply(geo2kegg, function(d) d[1:1000,])

# preprocessing and DE analysis for two of the datasets
geo2kegg <- maPreproc(geo2kegg[2:3])
geo2kegg <- runDE(geo2kegg)

# getting a subset of human KEGG gene sets
gs.file <- system.file("extdata/hsa_kegg_gs.gmt", package="EnrichmentBrowser")
kegg.gs <- EnrichmentBrowser::getGenesets(gs.file)

# applying two methods to two datasets
res <- runEA(geo2kegg, methods=c("ora", "camera"), gs=kegg.gs, perm=0)
```

Index