Package: MIRit (via r-universe)

November 1, 2026

Title Integrate microRNA and gene expression to decipher pathway complexity

Version 1.8.0

Date 2023-11-23

Description MIRit is an R package that provides several methods for investigating the relationships between miRNAs and genes in different biological conditions. In particular, MIRit allows to explore the functions of dysregulated miRNAs, and makes it possible to identify miRNA-gene regulatory axes that control biological pathways, thus enabling the users to unveil the complexity of miRNA biology. MIRit is an all-in-one framework that aims to help researchers in all the central aspects of an integrative miRNA-mRNA analyses, from differential expression analysis to network characterization.

License GPL (>= 3)

URL https://jacopo-ronchi.github.io/MIRit/, https://github.com/jacopo-ronchi/MIRit

BugReports https://github.com/jacopo-ronchi/MIRit/issues

biocViews Software, GeneRegulation, NetworkEnrichment, NetworkInference, Epigenetics, FunctionalGenomics, SystemsBiology, Network, Pathways, GeneExpression, DifferentialExpression

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports AnnotationDbi, BiocFileCache, BiocParallel, DESeq2, edgeR, fgsea, genekitr, geneset, ggplot2, ggpubr, graph, graphics, graphite, grDevices, httr, limma, methods, Rcpp, Rgraphviz (>= 2.44.0), rlang, stats, utils

Collate 'AllGenerics.R' 'AllClasses.R' 'MIRit-package.R' 'RcppExports.R' 'association.R' 'batch-correction.R' 'data.R'

'differential-expression.R' 'enrichment.R' 'integration.R' 'show-methods.R' 'targets.R' 'topological-integration.R' 'utils.R' 'visualization.R'

Suggests BiocStyle, biomaRt, BSgenome.Hsapiens.UCSC.hg38, GenomicRanges, ggrepel, ggridges, Gviz, gwasrapidd, knitr, MonoPoly, org.Hs.eg.db, rmarkdown, testthat (>= 3.0.0)

Depends MultiAssayExperiment, R (>= 4.4.0)

LazyData false

VignetteBuilder knitr

Config/testthat/edition 3

LinkingTo Rcpp

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 13:02:17 UTC

RemoteUrl https://github.com/bioc/MIRit

RemoteRef RELEASE_3_23

RemoteSha 230c15d5b90b58f9f2e70939fbd6a1d4415d519a

Contents

addDifferentialExpression . . . 3
augmentedPathways . . . 6
batchCorrection . . . 7
deAccessors . . . 9
deAnalysis . . . 10
enrichedFeatures . . . 15
enrichGenes . . . 16
enrichmentBarplot . . . 20
enrichmentDatabase . . . 21
enrichmentDotplot . . . 22
enrichmentMethod . . . 23
enrichmentMetric . . . 24
enrichmentResults . . . 24
enrichTargets . . . 25
findMirnaSNPs . . . 27
FunctionalEnrichment-class . . . 29
geneCounts . . . 31
geneSet . . . 31
getEvidence . . . 32
getTargets . . . 33
gseaPlot . . . 36
gseaRidgeplot . . . 37
integratedPathways . . . 38

integration . . . 39
integrationDatabase . . . 40
integrationDotplot . . . 40
IntegrativePathwayAnalysis-class . . . 42
listPathways . . . 44
loadExamples . . . 45
mirnaCounts . . . 46
MirnaExperiment . . . 47
MirnaExperiment-class . . . 49
mirnaIntegration . . . 54
mirnaTargets . . . 57
mirVariantPlot . . . 58
pairedSamples . . . 60
plotCorrelation . . . 61
plotDE . . . 63
plotDimensions . . . 65
plotVolcano . . . 67
preparePathways . . . 69
searchDisease . . . 71
setTargets . . . 72
significantAccessors . . . 73
supportedOrganisms . . . 74
topologicalAnalysis . . . 75
visualizeNetwork . . . 78

Index **82**

`addDifferentialExpression` *Manually add differential expression results to a MirnaExperiment object*

Description

This function allows to add miRNA and gene differential expression results to a `MirnaExperiment` object. Instead of running `performMirnaDE()` and `performGeneDE()` functions, this one allows to use differential expression analyses carried out in other ways. Note that it is possible to manually add differential expression results just for miRNAs or just for genes. This is particularly useful in order to use the pipeline implemented in MIRit for proteomic data and for expression data deriving from different technologies.

Usage

```
addDifferentialExpression(
  mirnaObj,
  mirnaDE = NULL,
  geneDE = NULL,
  mirna.logFC = 0,
```

```
  mirna.pCutoff = 0.05,
  mirna.pAdjustment = "fdr",
  gene.logFC = 0,
  gene.pCutoff = 0.05,
  gene.pAdjustment = "fdr"
)
```

Arguments

mirnaObj	A MirnaExperiment object containing miRNA and gene data
mirnaDE	A data.frame containing the output of miRNA differential expression analysis. Check the *details* section to see the required format. Default is NULL not to add miRNA differential expression results
geneDE	A data.frame containing the output of gene differential expression analysis. Check the *details* section to see the required format. Default is NULL not to add gene differential expression results
mirna.logFC	The minimum log2 fold change required to consider a miRNA as differentially expressed. Optional, default is 0
mirna.pCutoff	The adjusted p-value cutoff to use for miRNA statistical significance. The default value is 0.05
mirna.pAdjustment	The p-value correction method for miRNA multiple testing. It must be one of: fdr (default), BH, none, holm, hochberg, hommel, bonferroni, BY
gene.logFC	The minimum log2 fold change required to consider a gene as differentially expressed. Optional, default is 0
gene.pCutoff	The adjusted p-value cutoff to use for gene statistical significance. The default value is 0.05
gene.pAdjustment	The p-value correction method for gene multiple testing. It must be one of: fdr (default), BH, none, holm, hochberg, hommel, bonferroni, BY

Details

The following paragraphs briefly explain the formats needed for mirnaDE, geneDE, and differential expression parameters.

mirnaDE and geneDE:

mirnaDE and geneDE are two objects of class data.frame containing the results of miRNA and gene differential expression analysis respectively. These tables should contain the differential expression results for all miRNAs/genes analyzed, not just for statistically significant species.

Note that you can individually add differential expression results for miRNAs and genes. For instance, it is possible to manually add gene differential expression through this function, while performing miRNA differential expression through the performMirnaDE() function, and vice versa. In order to only add miRNA or gene differential expression results, you must leave mirnaDE or geneDE slots to NULL.

All data.frame objects can be used, as long as they have:

- One column containing miRNA/gene names (according to miRBase/hgnc nomenclature). Accepted column names are: `ID`, `Symbol`, `Gene_Symbol`, `Mirna`, `mir`, `Gene`, `gene.symbol`, `Gene.symbol`;
- One column with log2 fold changes. Accepted column names are: `logFC`, `log2FoldChange`, `FC`, `lFC`;
- One column with average expression. Accepted column names are: `AveExpr`, `baseMean`, `logCPM`;
- One column with the p-values resulting from the differential expression analysis. Accepted column names are: `P.Value`, `pvalue`, `PValue`, `Pvalue`;
- One column containing p-values adjusted for multiple testing. Accepted column names are: `adj.P.Val`, `padj`, `FDR`, `fdr`, `adj`, `adj.p`, `adjp`.

Differential expression cutoffs:

`mirna.logFC`, `mirna.pCutoff`, `mirna.pAdjustment`, and `gene.logFC`, `gene.pCutoff`, `gene.pAdjustment` represent the parameters used to define the significance of differential expression results. These are needed in order to inform MIRit about the features that are considered as differentially expressed.

Value

A `MirnaExperiment` object containing differential expression results. To access these results, the user may run the `mirnaDE()` and `geneDE()` functions for miRNAs and genes, respectively.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example data
data(geneCounts, package = "MIRit")
data(mirnaCounts, package = "MIRit")

# create samples metadata
meta <- data.frame(
    "primary" = colnames(geneCounts),
    "mirnaCol" = colnames(mirnaCounts), "geneCol" = colnames(geneCounts),
    "disease" = c(rep("PTC", 8), rep("NTH", 8)),
    "patient" = c(rep(paste("Sample_", seq(8), sep = ""), 2))
)

# create a 'MirnaExperiment' object
obj <- MirnaExperiment(
    mirnaExpr = mirnaCounts, geneExpr = geneCounts,
    samplesMetadata = meta, pairedSamples = TRUE
)

# load example tables with differential expression results
de_m <- mirnaDE(object = loadExamples(), onlySignificant = FALSE)
de_g <- geneDE(object = loadExamples(), onlySignificant = FALSE)
```

```
# add DE results to MirnaExperiment object
obj <- addDifferentialExpression(obj, de_m, de_g,
    mirna.pCutoff = 0.05,
    gene.pCutoff = 0.05
)
```

---

augmentedPathways *Access the miRNA-augmented pathways that were used during TAIPA*

Description

This function accesses the pathways slot of a FunctionalEnrichment object and returns a list object with the augmented pathways that were considered by the topologicalAnalysis() function to perform the integrative analysis.

Usage

```
augmentedPathways(object)
```

Arguments

object	An object of class IntegrativePathwayAnalysis containing the results of a miRNA-mRNA pathway analysis

Value

A list object with the miRNA-augmented biological pathways.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load the example IntegrativePathwayAnalysis object
obj <- loadExamples("IntegrativePathwayAnalysis")

# extract the pathways
ps <- augmentedPathways(obj)
```

batchCorrection *Correct for batch effects in miRNA and gene expression measurements*

Description

This function allows to remove unwanted batch effects from miRNA and gene expression matrices. In particular, this function fits a linear model to miRNA/gene expression levels, and then removes the variability caused by batch effects. Furthermore, a weighted surrogate variable analysis (WSVA) can also be included to remove the effects due to surrogate variables.

Usage

```
batchCorrection(
  mirnaObj,
  assay,
  batch = NULL,
  batch2 = NULL,
  covariates = NULL,
  includeWsva = FALSE,
  n.sv = 1L,
  weight.by.sd = TRUE
)
```

Arguments

mirnaObj	A MirnaExperiment object containing miRNA and gene data
assay	The expression matrix to correct. It must be one of genes and microRNA
batch	It must be the name of a variable present in the colData of a MirnaExperiment object (eg. "disease"), or, alternatively, it must be a character/factor object that defines batch memberships. See the **details** section for additional information
batch2	It must be the name of a variable present in the colData of a MirnaExperiment object (eg. "disease"), or, alternatively, it must be a character/factor object that defines another series of batches that have additive effects to those specified in batch. See the **details** section for additional information
covariates	Additional numeric covariates that we want to correct for. It must be a character vector containing the names of numeric variables present in the colData of a MirnaExperiment object (eg. c("age", "RIN", "quantity")), or, alternatively, it must be a simple matrix object. See the **details** section for additional information
includeWsva	Logical, whether to correct for surrogate variables or not. Default is FALSE
n.sv	The number of surrogate variables to estimate
weight.by.sd	Logical, whether to specifically tune the surrogate variables to the more variable genes or not. Default is TRUE

Details

Batch effects consist in unwanted sources of technical variation that confound expression variability and limit downstream analyses. Since the reliability of biological conclusions of integrative miRNA-mRNA analyses depends on the association between miRNA and gene expression levels, it is pivotal to ensure that expression measurements are not affected by technical variations. To account for effects that influence both gene and miRNA expression (e.g. sex, smoking, hospitals), the most appropriate way is to perform a partial correlation analysis by including `partial = TRUE` when calling the `mirnaIntegration()` function. Nevertheless, for effects that are impacting either gene or miRNA expression matrix, the only option is to regress out their influence using this function before using the `mirnaIntegration()` function to perform a correlation analysis.

Usually, given a `MirnaExperiment` object, the user should specify:

- the `assay` from which we want to remove batch effects (one between `genes` and `microRNA`);
- the `batch` variable, which is a variable that defines the different batches;
- the `batch2` variable, which can be included to correct for a second series of batches that have additive effects to those specified in `batch`;
- the `covariates` variables, which allows correction for one or more continuous numeric effects.

In particular, `batch` and `batch2` could be provided as the names of covariates included in the `colData` of a `MirnaExperiment` object. Alternatively, they can be `character`/`factor` objects that declare batch memberships. Similarly, `covariates` can be supplied as a vector containing the names of numeric variables listed in the `colData` of `MirnaExperiment` objects, or they can be provided as a simple `matrix`.

Additionally, the influence of unknown sources of technical variation can be removed by including surrogate variables estimated through WSVA. To do so, we can set `includeWsva` to TRUE, and then we can specify the number of surrogate variables to use through the `n.sv` parameter. Further, the surrogate variables can be tuned to the more variable genes by setting `weight.by.sd` to TRUE.

Please note that we only recommend to remove batch effects directly from expression measurements prior to correlation analysis. This function can't be used to remove batch effects before differential expression analysis, because for that purpose, it is better to include batch variables in the linear model. In this way, we do not underestimate the residual degrees of freedom, so that the calculated standard errors, t-statistics and p-values are not overoptimistic.

Value

A `MirnaExperiment` object containing batch effect-corrected expression matrices.

Note

To estimate surrogate variables and to remove batch effects from expression data, MIRit uses the `limma::wsva()` and `limma::removeBatchEffect()` functions, respectively.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK (2015). "limma powers differential expression analyses for RNA-sequencing and microarray studies." Nucleic Acids Research, 43(7), e47. doi:10.1093/nar/gkv007.

Ronchi, J., & Foti, M. (2026). MIRit: An integrative R framework for the identification of impaired miRNA–mRNA regulatory networks in complex diseases. Bioinformatics Advances, vbag042. https://doi.org/10.1093/bioadv/vbag042

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# correct batch effects due to the patient from miRNA expression matrix
obj <- batchCorrection(obj, "microRNA", batch = "patient")
```

`deAccessors` *Extract differentially expressed miRNAs and genes from a* `MirnaExperiment` *object*

Description

The `mirnaDE()` and `geneDE()` are two accessor functions for the `mirnaDE` and `geneDE` slots of `MirnaExperiment` class, respectively. Thus, they can be used to explore the results of miRNA and gene differential expression analysis stored in a `MirnaExperiment` object.

Usage

```
mirnaDE(object, onlySignificant = TRUE, param = FALSE, returnObject = FALSE)

geneDE(object, onlySignificant = TRUE, param = FALSE, returnObject = FALSE)
```

Arguments

`object`	A `MirnaExperiment` object containing miRNA and gene data
`onlySignificant`	Logical, if `TRUE` differential expression results will be returned just for statistically significant miRNAs/genes, if `FALSE` the full table of miRNA/gene differential expression will be provided. Default is `TRUE` to only report significant miRNAs/genes
`param`	Logical, whether to return the complete `list` object with the parameters used, or just the results stored in `data`. Default is FALSE
`returnObject`	Logical, if `TRUE` this function will return the limma/edgeR/DESeq2 object used for differential expression analysis

Value

A `data.frame` with miRNA/gene differential expression, or a `list` object with the parameters used if `param = TRUE`.

Functions

- `mirnaDE()`: Extract miRNA differential expression results
- `geneDE()`: Extract gene differential expression results

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# access miRNA differential expression of a MirnaExperiment object
sig <- mirnaDE(obj)
all <- mirnaDE(obj, onlySignificant = FALSE)

# access gene differential expression of a MirnaExperiment object
sig <- geneDE(obj)
all <- geneDE(obj, onlySignificant = FALSE)
```

---

deAnalysis *Perform differential expression analysis*

Description

`performMirnaDE()` and `performGeneDE()` are two functions provided by MIRit to conduct miRNA and gene differential expression analysis, respectively. In particular, these functions allow the user to compute differential expression through different methods, namely `edgeR` (Quasi-Likelihood framework), `DESeq2`, `limma-voom` and `limma`. Data deriving from NGS experiments and microarray technology are all suitable for these functions. For precise indications about how to use these functions, please refer to the *details* section.

Usage

```
performMirnaDE(
  mirnaObj,
  group,
  contrast,
  design,
  method = "edgeR",
  logFC = 0,
```

```
  pCutoff = 0.05,
  pAdjustment = "fdr",
  filterByExpr.args = list(),
  calcNormFactors.args = list(),
  estimateDisp.args = list(robust = TRUE),
  glmQLFit.args = list(),
  glmQLFTest.args = list(),
  DESeq.args = list(),
  useVoomWithQualityWeights = TRUE,
  voom.args = list(),
  lmFit.args = list(),
  eBayes.args = list(),
  useArrayWeights = TRUE,
  useWsva = FALSE,
  wsva.args = list(),
  arrayWeights.args = list(),
  useDuplicateCorrelation = FALSE,
  correlationBlockVariable = NULL,
  duplicateCorrelation.args = list()
)

performGeneDE(
  mirnaObj,
  group,
  contrast,
  design,
  method = "edgeR",
  logFC = 0,
  pCutoff = 0.05,
  pAdjustment = "fdr",
  filterByExpr.args = list(),
  calcNormFactors.args = list(),
  estimateDisp.args = list(robust = TRUE),
  glmQLFit.args = list(),
  glmQLFTest.args = list(),
  DESeq.args = list(),
  useVoomWithQualityWeights = TRUE,
  voom.args = list(),
  lmFit.args = list(),
  eBayes.args = list(),
  useArrayWeights = TRUE,
  useWsva = FALSE,
  wsva.args = list(),
  arrayWeights.args = list(),
  useDuplicateCorrelation = FALSE,
  correlationBlockVariable = NULL,
  duplicateCorrelation.args = list()
)
```

Arguments

mirnaObj	A MirnaExperiment object containing miRNA and gene data
group	The variable of interest for differential expression analysis. It must be the column name of a variable present in the metadata (colData) of a MirnaExperiment object. See the *details* section for additional information
contrast	A character object that specifies the groups to be compared during differential expression analysis, separated by a dash (e.g. 'disease-healthy'). Note that reference group must be the last one, for additional information see the *details* section
design	An R formula that indicates the model to fit. It must include the variable of interest (group) together with eventual covariates (e.g. '~ 0 + disease + sex'). Please note that group variable must be the first one. See the *details* section for additional information
method	The statistical package used to compute differential expression. For NGS experiments, it must be one of edgeR (default), DESeq2, and voom (for limma-voom). Instead, for microarray data, only limma can be used
logFC	The minimum log2 fold change required to consider a gene as differentially expressed. Optional, default is 0
pCutoff	The adjusted p-value cutoff to use for statistical significance. The default value is 0.05
pAdjustment	The p-value correction method for multiple testing. It must be one of: fdr (default), BH, none, holm, hochberg, hommel, bonferroni, BY
filterByExpr.args	A list object containing additional arguments passed to edgeR::filterByExpr() function. It is used when method is set to edgeR or voom
calcNormFactors.args	A list object containing additional arguments passed to edgeR::calcNormFactors() function. It is used when method is set to edgeR or voom
estimateDisp.args	A list object containing additional arguments passed to edgeR::estimateDisp() function. It is used when method is set to edgeR. Default is list(robust = TRUE) to use the robust parameter
glmQLFit.args	A list object containing additional arguments passed to edgeR::glmQLFit() function. It is used when method is set to edgeR
glmQLFTest.args	A list object containing additional arguments passed to edgeR::glmQLFTest() function. It is used when method is set to edgeR
DESeq.args	A list object containing additional arguments passed to DESeq2::DESeq() function. It is used when method is set to DESeq
useVoomWithQualityWeights	Logical, whether to use the limma::voomWithQualityWeights() function or just the limma::voom() function. It is used when method is set to voom. Default is TRUE

voom.args A list object containing additional arguments passed to limma::voom() function or limma::voomWithQualityWeights() function. It is used when method is set to voom

lmFit.args A list object containing additional arguments passed to limma::lmFit() function. It is used when method is set to voom or limma

eBayes.args A list object containing additional arguments passed to limma::eBayes() function. It is used when method is set to voom or limma

useArrayWeights
Logical, whether to use the limma::arrayWeights() function or not. It is used when method is set to limma. Default is TRUE

useWsva Logical, whether to use the limma::wsva() function or not. It is used when method is set to limma. Default is FALSE

wsva.args A list object containing additional arguments passed to limma::wsva() function. It is used when method is set to limma

arrayWeights.args
A list object containing additional arguments passed to limma::arrayWeights() function. It is used when method is set to limma

useDuplicateCorrelation
Logical, whether to use the limma::duplicateCorrelation() function or not. It is used when method is set to limma. Default is FALSE

correlationBlockVariable
It is the blocking variable to use for limma::duplicateCorrelation(). Default is NULL

duplicateCorrelation.args
A list object containing additional arguments passed to limma::duplicateCorrelation() function. It is used when method is set to limma

Details

When performing differential expression for NGS experiments, count matrices are detected and method parameter must be one of edgeR, DESeq2, and voom. On the other hand, when dealing with microarray studies, only limma can be used.

To calculate differential expression, MIRit must be informed about the variable of interest and the desired contrast. In particular, the group parameter must be the name of a variable present in the metadata (colData) of a MirnaExperiment object, which specifies the variable used to compute differential expression analysis, between the groups indicated in contrast. Specifically, contrast must be a character vector that defines the levels to compare separated by a dash. For example, if we have a variable named 'condition', with two levels, namely 'disease' and 'healthy', we can identify differentially expressed genes in 'disease' samples compared to 'healthy' subjects by specifying: group = 'condition' and contrast = 'disease-healthy'. Furthermore, the user needs to specify the model to fit expression values. To do so, the user has to state the model formula in the design parameter. Please note that for a correct inner working of these functions, the group variable of interest must be the *first* variable in model formula. Moreover, the user can include in the design any other sources of variation by specifying covariates that will be taken into account. For instance, if we want to compare 'disease' subjects against 'healthy' individuals, without the influence of sex differences, we may specify design = ~ condition + sex, where 'sex' is also a variable present in the metadata (colData) of mirnaObj.

Notably, for all the methods available, the user can supply additional arguments to the functions implemented in edgeR, DESeq2 and limma. Therefore, the user has finer control over how the differential expression analysis is performed. In this regard, for microarray studies, the user may opt to include weighted surrogate variable analysis (WSVA) to correct for unknown sources of variation (useWsva = TRUE). Moreover, for microarray data, the arrayWeights() function in limma can be used to assess differential expression with respect to array qualities. Also, the duplicateCorrelation() function in limma may be included in the pipeline in order to block the effect of correlated samples. To do this, the user must set useDuplicateCorrelation = TRUE, and must specify the blocking variable through the correlationBlockVariable parameter. Additionally, when using limma-voom, the user may estimate voom transformation with or without quality weights (by specifying useVoomWithQualityWeights = TRUE).

Value

A MirnaExperiment object containing differential expression results. To access these results, the user may run the mirnaDE() and geneDE() functions for miRNAs and genes, respectively.

Functions

- performMirnaDE(): Perform differential expression analysis for miRNAs
- performGeneDE(): Perform differential expression analysis for genes

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK (2015). "limma powers differential expression analyses for RNA-sequencing and microarray studies." Nucleic Acids Research, 43(7), e47. doi:10.1093/nar/gkv007.

Law, CW, Chen, Y, Shi, W, and Smyth, GK (2014). "Voom: precision weights unlock linear model analysis tools for RNA-seq read counts". Genome Biology 15, R29

Robinson MD, McCarthy DJ, Smyth GK (2010). "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data." Bioinformatics, 26(1), 139-140. doi:10.1093/bioinformatics/btp616.

Love MI, Huber W, Anders S (2014). "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2." Genome Biology, 15, 550. doi:10.1186/s13059-014-0550-8.

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# perform miRNA DE with edgeR
obj <- performMirnaDE(obj,
    group = "disease", contrast = "PTC-NTH",
    design = ~ 0 + disease + patient, method = "edgeR"
)
```

```
# perform miRNA DE with DESeq2
obj <- performMirnaDE(obj,
    group = "disease", contrast = "PTC-NTH",
    design = ~ 0 + disease + patient, method = "DESeq2"
)

# perform miRNA DE with limma-voom
obj <- performMirnaDE(obj,
    group = "disease", contrast = "PTC-NTH",
    design = ~ 0 + disease + patient, method = "voom"
)
```

`enrichedFeatures` *Extract the names of the pre-ranked features in a GSEA experiment*

Description

This function accesses the `features` slot of a `FunctionalEnrichment` object and returns a `character` vector with the names of the features considered in GSEA in the same order as the ranking metric.

Usage

```
enrichedFeatures(object)
```

Arguments

`object`	An object of class `FunctionalEnrichment` containing enrichment results

Value

A `character` vector with the names of the genes ordered based on the ranking metric.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example FunctionalEnrichment object
obj <- loadExamples("FunctionalEnrichment")

# extract the ranking metric
rmet <- enrichmentMetric(obj)

## extract the corresponding names
rnames <- enrichedFeatures(obj)
```

enrichGenes *Perform functional enrichment analysis of genes*

Description

This function allows to investigate the biological functions and pathways that result dysregulated across biological conditions. In particular, different enrichment approaches can be used, including over-representation analysis (ORA), gene-set enrichment analysis (GSEA), and Correlation Adjusted MEan RAnk gene set test (CAMERA). Moreover, for all these analyses, the enrichment can be carried out using different databases, namely Gene Ontology (GO), Kyoto Encyclopedia of Genes and Genomes (KEGG), MsigDB, WikiPathways, Reactome, Enrichr, Disease Ontology (DO), Network of Cancer Genes (NCG), DisGeNET, and COVID19. For exhaustive information on how to use this function, please refer to the *details* section.

Usage

```
enrichGenes(
  mirnaObj,
  method = "GSEA",
  database = "GO",
  category = NULL,
  organism = "Homo sapiens",
  pCutoff = 0.05,
  pAdjustment = "fdr",
  minSize = 10L,
  maxSize = 500L,
  rankMetric = "signed.pval",
  eps = 1e-50
)
```

Arguments

mirnaObj	A MirnaExperiment object containing miRNA and gene data
method	The functional enrichment analysis to perform. It must be one of ORA, GSEA (default), and CAMERA. For additional information, see the *details* section
database	The name of the database used for the enrichment analysis. It must be one of: GO, KEGG, MsigDB, WikiPathways, Reactome, Enrichr, DO, NCG, DisGeNET, COVID19. Default is GO
category	The desired subcategory of gene sets present in database. Please, see the *details* section to check the available categories for each database. Default is NULL to use default categories
organism	The name of the organism under consideration. The different databases have different supported organisms. To see the list of supported organisms for a given database, use the supportedOrganisms() function. Default is Homo sapiens
pCutoff	The adjusted p-value cutoff to use for statistical significance. The default value is 0.05

pAdjustment	The p-value correction method for multiple testing. It must be one of: fdr (default), BH, none, holm, hochberg, hommel, bonferroni, BY
minSize	The minimum size for a gene set. All gene sets containing less than this number of genes will not be considered. Default is 10
maxSize	The maximum size for a gene set. All gene sets containing more than this number of genes will not be considered. Default is 500
rankMetric	The ranking statistic used to order genes before performing GSEA. It must be one of signed.pval (default), logFC, and log.pval. For additional information, refer to the *details* section
eps	The lower boundary for p-value calculation (default is 1e-50). To compute exact p-values, this parameter can be set to 0, even though the analysis will be slower

Details

Enrichment method:

The method used for functional enrichment analysis will drastically influence the biological results, and thus, it must be carefully chosen. ORA (Boyle et al., 2004) takes differentially expressed genes (separately considering upregulated and downregulated features) and uses the hypergeometric test to infer the biological processes that are regulated by these genes more than would be expected by chance. The downside of this approach is that we only consider genes that passed a pre-defined threshold, thus losing all the slight changes in gene expression that may have important biological consequences.

To address this limit, GSEA was introduced (Subramanian, 2005). This analysis starts by ranking genes according to a specific criterion, and then uses a running statistic that is able to identify even slight but coordinated expression changes of genes belonging to a specific pathway. Therefore, GSEA is the default method used in MIRit to perform the functional enrichment analysis of genes.

Moreover, in addition to ORA and GSEA, this function allows to perform the enrichment analysis through CAMERA (Wu and Smyth, 2012), which is another competitive test used for functional enrichment of genes. The main advantage of this method is that it adjusts the gene set test statistic according to inter-gene correlations. This is particularly interesting since it was demonstrated that inter-gene correlations may affect the reliability of functional enrichment analyses.

Databases and categories:

Regarding gene sets, multiple databases can be used to investigate the consequences of gene expression alterations. However, different databases also includes several subcategories with different annotations. To specifically query desired categories, the category parameter is used. As a reference, here are listed the available categories for the different databases supported:

- Gene Ontology (GO):
  - bp, for GO - Biological Processes;
  - mf, for GO - Molecular Function;
  - cc, for GO - Cellular Component;
- Kyoto Encyclopedia of Genes and Genomes (KEGG):
  - pathway, for KEGG biological pathways;
  - module, for KEGG reaction modules;
  - enzyme, for KEGG enzyme nomenclature;
  - disease, for KEGG diseases (only Homo sapiens supported);

  - drug, for KEGG drug targets (only Homo sapiens supported);
  - network, for KEGG disease/drug perturbation netowrks (only Homo sapiens supported);
- MsigDB:
  - H, for MsigDB hallmark genes of specific biological states/processes;
  - C1, for gene sets of human chromosome cytogenetic bands;
  - C2-CGP, for expression signatures of genetic and chemical perturbations;
  - C2-CP-BIOCARTA, for canonical pathways gene sets derived from the BioCarta pathway database;
  - C2-CP-KEGG, for canonical pathways gene sets derived from the KEGG pathway database;
  - C2-CP-PID, for canonical pathways gene sets derived from the PID pathway database;
  - C2-CP-REACTOME, for canonical pathways gene sets derived from the Reactome pathway database;
  - C2-CP-WIKIPATHWAYS, for canonical pathways gene sets derived from the WikiPathways database;
  - C3-MIR-MIRDB, for gene sets containing high-confidence gene-level predictions of human miRNA targets as catalogued by miRDB v6.0 algorithm;
  - C3-MIR-MIR_Legacy, for older gene sets that contain genes sharing putative target sites of human mature miRNA in their 3'-UTRs;
  - C3-TFT-GTRD, for genes that share GTRD predicted transcription factor binding sites in the region -1000,+100 bp around the TSS for the indicated transcription factor;
  - C3-TFT-TFT_Legacy, for older gene sets that share upstream cis-regulatory motifs which can function as potential transcription factor binding sites;
  - C4-CGN, for gene sets defined by expression neighborhoods centered on 380 cancer-associated genes;
  - C4-CM, for cancer modules as defined by Segal et al. 2004;
  - C5-GO-BP, for GO - biological process ontology;
  - C5-GO-CC, for GO - cellular component ontology;
  - C5-GO-MF, for GO - molecular function ontology;
  - C5-HPO, for Human Phenotype ontology (HPO);
  - C6, for gene sets that represent signatures of cellular pathways which are often dis-regulated in cancer;
  - C7-IMMUNESIGDB, for manually curated gene sets representing chemical and genetic perturbations of the immune system;
  - C7-VAX, for gene sets deriving from the Human Immunology Project Consortium (HIPC) describing human transcriptomic immune responses to vaccinations;
  - C8, for gene sets that contain curated cluster markers for cell types;
- WikiPathways;
- Reactome;
- Enrichr:
  - All avaliable gene sets can be listed through geneset::enrichr_metadata
- Disease Ontology (DO);
- Network of Cancer Genes (NCG):
  - v6, for the sixth version;
  - v7, for the seventh version;

- DisGeNET;
- COVID-19.

Supported organisms:

For each database, different organisms are supported. To check the supported organisms for a given database, MIRit provides the `supportedOrganisms()` function.

GSEA ranking statistic:

The ranking statistic used to order genes before conducting GSEA is able to influence the biological interpretation of functional enrichment results. Several metrics have been used in scientific literature. MIRit implements the possibility of using `signed.pval`, `logFC`, and `log.pval`. In particular, the simplest option is to rank genes according to their `logFC` value. However, this procedure is biased by higher variance for lowly abundant genes.

Therefore, we recommend to use the `signed.pval` metric, which consists in the p-value of a gene multiplied for the sign of its logFC, i.e. `sign(logFC) * p-value`. Alternatively, `log,pval` metric, which consist in the product of logFC and p-value, i.e. `logFC * p-value` can also be used.

Value

For method `GSEA` and `CAMERA`, this function produces an object of class `FunctionalEnrichment` containing enrichment results. Instead, when `ORA` is used, this function returns a `list` object with two elements, namely 'upregulated' and 'downregulated', each containing a `FunctionalEnrichment` object storing enrichment results of upregulated and downregulated genes, respectively.

To access results of `FunctionalEnrichment` objects, the user can use the `enrichmentResults()` function. Additionally, MIRit provides several functions to graphically represent enrichment analyses, including `enrichmentBarplot()`, `enrichmentDotplot()`, `gseaPlot()`, and `gseaRidgeplot()`.

Note

To download gene sets from the above mentioned databases, MIRit uses the `geneset` R package. Moreover, to perform ORA and GSEA, MIRit implements the `fgsea` algorithm, whereas for CAMERA, the `limma` package is used.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Liu, Y., Li, G. Empowering biologists to decode omics data: the Genekitr R package and web server. BMC Bioinformatics 24, 214 (2023). https://doi.org/10.1186/s12859-023-05342-9.

Korotkevich G, Sukhov V, Sergushichev A (2019). "Fast gene set enrichment analysis." bioRxiv. doi:10.1101/060012, http://biorxiv.org/content/early/2016/06/20/060012.

Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK (2015). "limma powers differential expression analyses for RNA-sequencing and microarray studies." Nucleic Acids Research, 43(7), e47. doi:10.1093/nar/gkv007.

Wu D, Smyth GK. Camera: a competitive gene set test accounting for inter-gene correlation. Nucleic Acids Res. 2012 Sep 1;40(17):e133. doi: 10.1093/nar/gks461. Epub 2012 May 25. PMID: 22638577; PMCID: PMC3458527.

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# perform ORA with GO
de_enr <- enrichGenes(obj, method = "ORA", database = "GO")
```

`enrichmentBarplot` *Create a barplot for functional enrichment analysis*

Description

This function produces a barplot to show the results of functional enrichment analyses carried out through over-representation analysis (ORA), gene set enrichment analysis (GSEA), and competitive gene set test accounting for inter-gene correlation (CAMERA). In particular, this function can take as input enrichment results generated by the `enrichGenes()` function.

Usage

```
enrichmentBarplot(
  enrichment,
  showTerms = 10,
  showTermsParam = "ratio",
  splitDir = TRUE,
  title = NULL
)
```

Arguments

`enrichment`	An object of class `FunctionalEnrichment` containing enrichment results
`showTerms`	It is the number of terms to be shown, based on the order determined by the parameter `showTermsParam`; or, alternatively, a character vector indicating the terms to plot. Default is `10`
`showTermsParam`	The order in which the top terms are selected as specified by the `showTerms` parameter. It must be one of `ratio` (default), `padj`, `pval` and `overlap`
`splitDir`	Logical, if `TRUE` the resulting plot will be divided in two columns on the basis of enrichment direction (Up and Down). Default is `TRUE`. This only applies if enrichment method is GSEA or CAMERA
`title`	The title of the plot. Default is `NULL` not to include a plot title

Details

When producing a barplot with this function, significant pathways are ordered on the x-axis on the basis of the ratio between the number of overlapping genes in that set, and the total number of genes in the set. Moreover, the color scale of dots is relative to the adjusted p-values of each category.

Value

A ggplot graph with a barplot of enrichment results.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example FunctionalEnrichment object
obj <- loadExamples("FunctionalEnrichment")

# plot results
enrichmentBarplot(obj)
```

---

enrichmentDatabase *Access the database used for functional enrichment analyses*

Description

This function accesses the database slot of a FunctionalEnrichment object and returns a the name of the database used by the enrichGenes() function to perform the enrichment analysis.

Usage

```
enrichmentDatabase(object)
```

Arguments

object	An object of class FunctionalEnrichment containing enrichment results

Value

A character containing the name of the database, such as KEGG.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example FunctionalEnrichment object
obj <- loadExamples("FunctionalEnrichment")

# see the database
enrichmentDatabase(obj)
```

`enrichmentDotplot` *Create a dotplot for functional enrichment analysis*

Description

This function produces a dotplot to show the results of functional enrichment analyses carried out through over-representation analysis (ORA), gene set enrichment analysis (GSEA), and competitive gene set test accounting for inter-gene correlation (CAMERA). In particular, this function can take as input enrichment results generated by the `enrichGenes()` function.

Usage

```
enrichmentDotplot(
  enrichment,
  showTerms = 10,
  showTermsParam = "ratio",
  splitDir = TRUE,
  title = NULL
)
```

Arguments

`enrichment`	An object of class `FunctionalEnrichment` containing enrichment results
`showTerms`	It is the number of terms to be shown, based on the order determined by the parameter `showTermsParam`; or, alternatively, a character vector indicating the terms to plot. Default is `10`
`showTermsParam`	The order in which the top terms are selected as specified by the `showTerms` parameter. It must be one of `ratio` (default), `padj`, `pval` and `overlap`
`splitDir`	Logical, if `TRUE` the resulting plot will be divided in two columns on the basis of enrichment direction (Up and Down). Default is `TRUE`. This only applies if enrichment method is GSEA or CAMERA
`title`	The title of the plot. Default is `NULL` not to include a plot title

Details

When producing a dotplot with this function, significant pathways are ordered on the x-axis on the basis of the ratio between the number of overlapping genes in that set, and the total number of genes in the set. Moreover, the size of each dot is proportional to the number of overlapping features. Finally, the color scale of dots is relative to the adjusted p-values of each category.

Value

A `ggplot` graph with a dotplot of enrichment results.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example FunctionalEnrichment object
obj <- loadExamples("FunctionalEnrichment")

# plot results
enrichmentDotplot(obj)
```

`enrichmentMethod` *Access the method used for functional enrichment analyses*

Description

This function accesses the `method` slot of a `FunctionalEnrichment` object and returns a the name of the enrichment strategy used by the `enrichGenes()` function to perform the enrichment analysis.

Usage

```
enrichmentMethod(object)
```

Arguments

`object`	An object of class `FunctionalEnrichment` containing enrichment results

Value

A `character` containing the enrichment method, such as `GSEA`.

Author(s)

Jacopo Ronchi, `<jacopo.ronchi@unimib.it>`

Examples

```
# load example FunctionalEnrichment object
obj <- loadExamples("FunctionalEnrichment")

# see the method
enrichmentMethod(obj)
```

enrichmentMetric *Extract the GSEA ranking metric used for functional enrichment analyses*

Description

This function accesses the `statistic` slot of a `FunctionalEnrichment` object and returns a `numeric` vector with the metric used to rank genes in GSEA.

Usage

```
enrichmentMetric(object)
```

Arguments

`object`	An object of class `FunctionalEnrichment` containing enrichment results

Value

A `numeric` vector containing the ranking metric.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example FunctionalEnrichment object
obj <- loadExamples("FunctionalEnrichment")

# extract the ranking metric
rmet <- enrichmentMetric(obj)
```

enrichmentResults *Access the results of functional enrichment analyses*

Description

This function accesses the `data` slot of a `FunctionalEnrichment` object and returns a `data.frame` with enrichment results.

Usage

```
enrichmentResults(object)
```

Arguments

object	An object of class FunctionalEnrichment containing enrichment results

Value

A data.frame object containing the results of functional enrichment analyses, as returned by the enrichGenes() function.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example FunctionalEnrichment object
obj <- loadExamples("FunctionalEnrichment")

# extract results
de_df <- enrichmentResults(obj)
```

---

enrichTargets *Perform an enrichment analysis of integrated microRNA targets*

Description

This function allows to perform an over-representation analysis (ORA) of integrated miRNA targets in order to explore the biological effects of targets that are statistically associated/correlated with DE-miRNAs. The enrichment analysis can be performed using different databases, namely Gene Ontology (GO), Kyoto Encyclopedia of Genes and Genomes (KEGG), MsigDB, WikiPathways, Reactome, Enrichr, Disease Ontology (DO), Network of Cancer Genes (NCG), DisGeNET, and COVID19.

Usage

```
enrichTargets(
  mirnaObj,
  database = "GO",
  category = NULL,
  organism = "Homo sapiens",
  pCutoff = 0.05,
  pAdjustment = "fdr",
  minSize = 10L,
  maxSize = 500L
)
```

Arguments

mirnaObj	A MirnaExperiment object containing miRNA and gene data
database	The name of the database used for the enrichment analysis. It must be one of: GO, KEGG, MsigDB, WikiPathways, Reactome, Enrichr, DO, NCG, DisGeNET, COVID19. Default is GO
category	The desired subcategory of gene sets present in database. Please, see the *details* section to check the available categories for each database. Default is NULL to use default categories
organism	The name of the organism under consideration. The different databases have different supported organisms. To see the list of supported organisms for a given database, use the supportedOrganisms() function. Default is Homo sapiens
pCutoff	The adjusted p-value cutoff to use for statistical significance. The default value is 0.05
pAdjustment	The p-value correction method for multiple testing. It must be one of: fdr (default), BH, none, holm, hochberg, hommel, bonferroni, BY
minSize	The minimum size for a gene set. All gene sets containing less than this number of genes will not be considered. Default is 10
maxSize	The maximum size for a gene set. All gene sets containing more than this number of genes will not be considered. Default is 500

Details

For each database, different organisms are supported. To check the supported organisms for a given database, MIRit provides the supportedOrganisms() function.

Moreover, since different database support multiple subcategories, the category parameter can be set to specify the desired resource. For specific information regarding the available categories for the different databases, check the *details* section of the enrichGenes() documentation.

Value

This function produces a list object with two elements, namely 'upregulated' and 'downregulated', each containing a FunctionalEnrichment object storing enrichment results of upregulated and downregulated target genes, respectively.

To access results of FunctionalEnrichment objects, the user can use the enrichmentResults() function. Additionally, MIRit provides several functions to graphically represent enrichment analyses, including enrichmentBarplot(), and enrichmentDotplot().

Note

To download gene sets from the above mentioned databases, MIRit uses the geneset R package. Moreover, to perform ORA, MIRit implements the fgsea package in Bioconductor.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Liu, Y., Li, G. Empowering biologists to decode omics data: the Genekitr R package and web server. BMC Bioinformatics 24, 214 (2023). https://doi.org/10.1186/s12859-023-05342-9.

Korotkevich G, Sukhov V, Sergushichev A (2019). "Fast gene set enrichment analysis." bioRxiv. doi:10.1101/060012, http://biorxiv.org/content/early/2016/06/20/060012.

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# perform enrichment analysis of integrated targets with DO
targets_enrichment <- enrichTargets(obj, database = "DO")
```

---

findMirnaSNPs *Find disease-associated SNPs occurring at DE-miRNA loci*

Description

This function allows to identify disease-associated genomic variants affecting differentially expressed miRNA genes or their host genes. To do so, this function uses `gwasrapidd` to retrieve SNPs-disease associations, and then retains only SNPs that affect DE-miRNA genes or their relative host genes (for intronic miRNAs).

Usage

```
findMirnaSNPs(mirnaObj, diseaseEFO)
```

Arguments

`mirnaObj`	A `MirnaExperiment` object containing miRNA and gene data
`diseaseEFO`	The EFO identifier of a disease of interest. This can be identified with the `searchDisease()` function

Details

SNPs occurring within miRNAs may have important effects on the biological function of these transcripts. Indeed, a SNP present within a miRNA gene might alter its expression or the spectrum of miRNA targets.

To retrieve disease-SNPs, this function uses `gwasrapidd` package, which directly queries the NHGRI-EBI Catalog of published genome-wide association studies. After running this function, the user can use the `mirVariantPlot()` function to produce a trackplot for visualizing the genomic location of SNPs within miRNA genes.

Value

A `data.frame` containing details about disease-SNPs and the associated differentially expressed miRNAs:

- `variant` contains SNP identifiers;
- `gene` defines the gene affected by a disease-SNP (it may be the miRNA gene itself or the host gene of an intronic miRNA);
- `miRNA.gene` specifies the DE-miRNA gene present;
- `miRNA.precursor` specifies the name of the miRNA precursor affected by disease-SNPs;
- `chr` indicates the chromosome of SNPs;
- `position` shows the SNP position;
- `allele` displays possible alleles for this SNPs;
- `distance` specifies the distance between SNPs and miRNAs;
- `is_upstream` indicates whether SNP is upstream of miRNA gene;
- `is_downstream` indicates whether SNP is downstream of miRNA gene;
- `mirnaStrand` shows the strand;
- `mirnaStartPosition` displays the start position of DE-miRNA gene;
- `mirnaEndPosition` displays the end position of DE-miRNA gene.

Note

To retrieve disease-associated SNPs, this function makes use of the `gwasrapidd` package.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Ramiro Magno, Ana-Teresa Maia, gwasrapidd: an R package to query, download and wrangle GWAS catalog data, Bioinformatics, Volume 36, Issue 2, January 2020, Pages 649–650, https://doi.org/10.1093/bioinformatics/btz605

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()


# search disease
searchDisease("response to antidepressant")
disId <- "response to antidepressant"

# retrieve associated SNPs
association <- findMirnaSNPs(obj, disId)
```

FunctionalEnrichment-class
The FunctionalEnrichment *class*

Description

This class introduces the possibility to store the results of functional enrichment analyses such as over-representation analysis (ORA), gene set enrichment analysis (GSEA), and competitive gene set test accounting for inter-gene correlation (CAMERA). The different slots contained in this class are used to store enrichment results generated through the enrichGenes() function.

Usage

```
## S4 method for signature 'FunctionalEnrichment'
enrichmentResults(object)

## S4 method for signature 'FunctionalEnrichment'
enrichmentDatabase(object)

## S4 method for signature 'FunctionalEnrichment'
enrichmentMethod(object)

## S4 method for signature 'FunctionalEnrichment'
geneSet(object)

## S4 method for signature 'FunctionalEnrichment'
enrichmentMetric(object)

## S4 method for signature 'FunctionalEnrichment'
enrichedFeatures(object)

## S4 method for signature 'FunctionalEnrichment'
show(object)
```

Arguments

object	An object of class FunctionalEnrichment containing enrichment results

Value

- the enrichmentResults() function returns a data.frame object with the results of the functional enrichment analysis.
- the enrichmentDatabase() function returns a character that specifies the database that provided the gene-set collection.
- the enrichmentMethod() function returns a character with the enrichment method used for the analysis.

- the geneSet() function returns a list with the gene-set collections retrieved.
- the enrichmentMetric() function returns a numeric object with ranked gene statistic used for GSEA.
- the enrichmentMetric() function returns a character object with the names of the ranked features used for GSEA.

Functions

- enrichmentResults(FunctionalEnrichment): Access the data slot to take a closer look at all the enriched terms of an enrichment analysis
- enrichmentDatabase(FunctionalEnrichment): See the database used for the functional enrichment
- enrichmentMethod(FunctionalEnrichment): Visualize the approach used for the functional enrichment analysis
- geneSet(FunctionalEnrichment): Access the geneSet slot to see the collection of gene sets used for GSEA
- enrichmentMetric(FunctionalEnrichment): View the ranking metric used for GSEA
- enrichedFeatures(FunctionalEnrichment): View the names of the pre-ranked features used for GSEA
- show(FunctionalEnrichment): Show method for objects of class FunctionalEnrichment

Slots

data A data.frame object holding the output of enrichment analysis

method The method used to perform functional enrichment analysis (e.g. Gene Set Enrichment Analysis (GSEA))

organism The name of the organism under consideration (e.g. Homo sapiens)

database The name of the database used for the enrichment analysis (e.g. KEGG)

pCutoff A numeric value defining the threshold used for statistical significance in the enrichment analysis (e.g. 0.05)

pAdjustment A character indicating the method used to correct p-values for multiple testing (e.g. fdr)

features A character vector containing the list of features used for the enrichment

statistic A numeric vector containing the statistic used to run GSEA. This parameter is empty for ORA and CAMERA

universe The background universe of features. Typically, this is equal to the complete list of features assayed. This slot is NULL for GSEA

geneSet The gene set used for the functional enrichment analysis. It is a list object where each element contains the list of genes belonging to a specific pathway.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

geneCounts *Count matrix for gene expression in thyroid cancer*

Description

This dataset contains a gene expression matrix resulting from an RNA-Seq analysis of thyroid cancer. Specifically, these data originate from Riesco-Eizaguirre et al (2015), where they sequenced 8 papillary thyroid carcinomas (PTC) together with paired samples of normal thyroid tissue. The same thing was done for microRNAs in order to investigate the effects on target genes. Data included in this package have been obtained through the Gene Expression Omnibus (GEO accession: GSE63511).

Usage

```
data(geneCounts)
```

Format

`geneCounts`**:**

A `matrix` object containing samples as columns and genes as rows.

Value

A `matrix` object with 23710 rows and 16 columns.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE63511

References

Garcilaso Riesco-Eizaguirre et al., "The MiR-146b-3p/PAX8/NIS Regulatory Circuit Modulates the Differentiation Phenotype and Function of Thyroid Cells during Carcinogenesis," Cancer Research 75, no. 19 (September 30, 2015): 4119–30, https://doi.org/10.1158/0008-5472.CAN-14-3547.

geneSet *Extract the gene-sets used for functional enrichment analyses*

Description

This function accesses the `geneSet` slot of a `FunctionalEnrichment` object and returns a `list` with the collection of genes used for the enrichment.

Usage

```
geneSet(object)
```

Arguments

object	An object of class FunctionalEnrichment containing enrichment results

Value

A list containing the gene-sets.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example FunctionalEnrichment object
obj <- loadExamples("FunctionalEnrichment")

# extract the gene-sets
gs <- geneSet(obj)
```

---

getEvidence *Get the scientific evidence for a particular disease-SNP association*

Description

This function returns the biomedical evidence that supports the association between a particular SNP and a phenotypic trait.

Usage

```
getEvidence(variant, diseaseEFO)
```

Arguments

variant	The SNP ID of a particular variant of interest (e.g. 'rs394581')
diseaseEFO	The EFO identifier of a disease of interest. This can be identified with the function searchDisease()

Value

A tbl_df dataframe containing information about literature evidences for a disease-SNP association.

Note

To retrieve evidences of disease-SNP association, this function makes use of the `gwasrapidd` package.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Ramiro Magno, Ana-Teresa Maia, gwasrapidd: an R package to query, download and wrangle GWAS catalog data, Bioinformatics, Volume 36, Issue 2, January 2020, Pages 649–650, https://doi.org/10.1093/bioinformatics/btz605.

Examples

```
# searchDisease("Alzheimer disease")
evidence <- getEvidence("rs2075650", diseaseEFO = "Alzheimer disease")
```

`getTargets` *Get microRNA targets*

Description

This function allows to obtain human miRNA-target interactions using two databases, namely miRTarBase v10, which contains experimentally validated interactions, and the microRNA Data Integration Portal (mirDIP) database, which aggregates miRNA target predictions from 24 different resources by using an integrated score inferred from different prediction metrics. In this way, as demonstrated by Tokar et al. 2018, mirDIP reports more accurate predictions compared to those of individual tools. However, for species other than Homo sapiens only validated interactions are returned, since mirDIP is only available for human miRNAs.

Usage

```
getTargets(
  mirnaObj,
  organism = "Homo sapiens",
  score = "High",
  includeValidated = TRUE,
  evidence = "all",
  local = NULL
)
```

Arguments

mirnaObj	A MirnaExperiment object containing miRNA and gene data
organism	The specie for which you are retrieving miRNA target genes. Available species are: Homo sapiens (default), Mus musculus, Rattus norvegicus, Arabidopsis thaliana, Bos taurus, Caenorhabditis elegans, Danio rerio, Drosophila melanogaster, Gallus gallus, Sus scrofa
score	The minimum mirDIP confidence score. It must be one of Very High, High (default), Medium, Low, which correspond to ranks among top 1%, top 5% (excluding top 1%), top 1/3 (excluding top 5%) and remaining predictions, respectively
includeValidated	Logical, whether to include validated interactions from miRTarBase or not. Default is TRUE in order to retrieve both predicted and validated targets. Note that for species other than Homo sapines only validated interactions are considered.
evidence	The support evidence required for miRTarBase validated interactions. The possible options are strong, to only include targets with strong experimental support, and all (default) to also include validated interactions with less strong evidence.
local	A data.frame with a local copy of miRTarBase csv file with validated miRNA-mRNA interactions. This parameter is optional and is set to NULL by default, in order to download the database from miRTarBase on the first use.

Details

To define miRNA target genes, we can consider both experimentally validated and computationally predicted interactions. Interactions of the former type are generally preferred, since they are corroborated by biomolecular experiments. However, they are often not sufficient, thus making it necessary to consider the predicted interactions as well. The downside of miRNA target prediction algorithms is the scarce extend of overlap existing between the different tools. To address this issue, several ensemble methods have been developed, trying to aggregate the predictions obtained by different algorithms. Initially, several researchers determined as significant miRNA-target pairs those predicted by more than one tool (intersection method). However, this method is not able to capture an important number of meaningful interactions. Alternatively, other strategies used to merge predictions from several algorithms (union method). Despite identifying more true relationships, the union method leads to a higher proportion of false discoveries. Therefore, other ensemble methods including mirDIP started using other statistics to rank miRNA-target predictions obtained by multiple algorithms. For additional information on mirDIP database and its ranking metric check Tokar et al. 2018 and Hauschild et al. 2023.

This function defines miRNA targets by considering both validated interactions present in miRTarBase (version 10), and predicted interactions identified by mirDIP. Please note that for species other than Homo sapiens, only miRTarBase interactions are available. Further, it is possible to include all validated miRNA-target interactions, or limit the retrieval to interactions with strong supporting evidence.

Value

A MirnaExperiment object containing miRNA targets stored in the targets slot. Results can be accessed with the mirnaTargets() function.

Note

To access mirDIP database at https://ophid.utoronto.ca/mirDIP/, this function directly use mirDIP API through R.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Tomas Tokar and others, mirDIP 4.1—integrative database of human microRNA target predictions, Nucleic Acids Research, Volume 46, Issue D1, 4 January 2018, Pages D360–D370, https://doi.org/10.1093/nar/gkx1144.

Anne-Christin Hauschild and others, MirDIP 5.2: tissue context annotation and novel microRNA curation, Nucleic Acids Research, Volume 51, Issue D1, 6 January 2023, Pages D217–D225, https://doi.org/10.1093/nar/gkac1070.

Hsi-Yuan Huang and others, miRTarBase update 2022: an informative resource for experimentally validated miRNA–target interactions, Nucleic Acids Research, Volume 50, Issue D1, 7 January 2022, Pages D222–D230, https://doi.org/10.1093/nar/gkab1079.

Ronchi, J., & Foti, M. (2026). MIRit: An integrative R framework for the identification of impaired miRNA–mRNA regulatory networks in complex diseases. Bioinformatics Advances, vbag042. https://doi.org/10.1093/bioadv/vbag042

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# retrieve targets
obj <- getTargets(mirnaObj = obj)

# access targets
tg <- mirnaTargets(obj)
```

gseaPlot *Create a GSEA plot that displays the running enrichment score (ES) for a given pathway*

Description

This function creates a classic enrichment plot to show the results of gene set enrichment analyses (GSEA). In particular, this function takes as input GSEA results originating from the `enrichGenes()` function, and returns a ggplot2::ggplot2 object with GSEA plot. In this kind of plots, the running enrichment score (ES) for a given pathway is shown on the y-axis, whereas gene positions in the ranked list are reported on the x-axis.

Usage

```
gseaPlot(
  enrichment,
  pathway,
  showTitle = TRUE,
  rankingMetric = FALSE,
  lineColor = "green",
  lineSize = 1,
  vlineColor = "red",
  vlineSize = 0.6
)
```

Arguments

`enrichment`	An object of class `FunctionalEnrichment` containing enrichment results
`pathway`	It must be the name of a significantly enriched term/pathway for which we want to produce a GSEA plot (e.g. 'Thyroid hormone synthesis')
`showTitle`	Logical, whether to add the name of the pathway/term as plot title. Default is TRUE
`rankingMetric`	Logical, whether to show the variations of the ranking metric below the plot. Default is FALSE
`lineColor`	It must be an R color name that specifies the color of the running score line. Default is `green`. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
`lineSize`	The line width of the running score line. Default is `1`
`vlineColor`	It must be an R color name that specifies the color of the vertical line indicating the enrichment score (ES). Default is `red`. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
`vlineSize`	The line width of the vertical line indicating the enrichment score (ES). Default is `0.6`

Value

An object of class ggplot containing the GSEA plot.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example FunctionalEnrichment object
obj <- loadExamples("FunctionalEnrichment")

# plot results
gseaPlot(obj, pathway = "Thyroid hormone synthesis")
```

---

gseaRidgeplot *Create a ridgeplot to display the results of GSEA analysis*

Description

This function creates a ridgeplot that is useful for showing the results of GSEA analyses. The output of this function is a plot where enriched terms/pathways found with enrichGenes() function are visualized on the basis of the ranking metric used for the analysis. The resulting areas represent the density of signed p-values, log2 fold changes, or log.p-values belonging to genes annotated to that category.

Usage

```
gseaRidgeplot(
  enrichment,
  showTerms = 10,
  showTermsParam = "padj",
  title = NULL
)
```

Arguments

enrichment	An object of class FunctionalEnrichment containing enrichment results
showTerms	It is the number of terms to be shown, based on the order determined by the parameter showTermsParam; or, alternatively, a character vector indicating the terms to plot. Default is 10
showTermsParam	The order in which the top terms are selected as specified by the showTerms parameter. It must be one of ratio, padj (default), pval and overlap
title	The title of the plot. Default is NULL not to include a plot title

Value

An object of class ggplot containing the ridgeplot of GSEA results.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example FunctionalEnrichment object
obj <- loadExamples("FunctionalEnrichment")

# plot results
gseaRidgeplot(obj)
```

---

integratedPathways *Access the results of integrative miRNA-mRNA pathway analyses*

Description

This function accesses the data slot of a FunctionalEnrichment object and returns a data.frame with the results of an integrative topological analysis carried out through the topologicalAnalysis() function.

Usage

```
integratedPathways(object)
```

Arguments

object	An object of class IntegrativePathwayAnalysis containing the results of a miRNA-mRNA pathway analysis

Value

A data.frame object containing the results of the topological analysis, as returned by the topologicalAnalysis() function.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load the example IntegrativePathwayAnalysis object
obj <- loadExamples("IntegrativePathwayAnalysis")

# extract results
taipaRes <- integratedPathways(obj)
```

`integration` *Explore the results of the integration analysis between miRNAs and genes*

Description

After performing the integration analysis between miRNA and gene expression values with the `mirnaIntegration()` function, the results are stored in the `integration` slot of a `MirnaExperiment` object and can be explored with this function.

Usage

```
integration(object, param = FALSE)
```

Arguments

`object`	A `MirnaExperiment` object containing miRNA and gene data
`param`	Logical, whether to return the complete `list` object with the parameters used, or just the results stored in `data`. Default is FALSE

Value

If `param` is FALSE, then this functions returns a `data.frame` object containing the results of the integration analysis. Otherwise, a `list` object including the parameters used for the analysis will be returned.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# visualize the results of correlation analysis
res <- integration(obj)
res
```

integrationDatabase *Extract the database used for integrative miRNA-mRNA pathway analyses*

Description

This function accesses the `database` slot of a `FunctionalEnrichment` object and returns the name of the database used by the `topologicalAnalysis()` function to perform the integrative topological analysis.

Usage

```
integrationDatabase(object)
```

Arguments

`object`	An object of class `IntegrativePathwayAnalysis` containing the results of a miRNA-mRNA pathway analysis

Value

A `character` object with the name of the database used by the `topologicalAnalysis()` function, such as `KEGG`.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load the example IntegrativePathwayAnalysis object
obj <- loadExamples("IntegrativePathwayAnalysis")

# see the database
integrationDatabase(obj)
```

integrationDotplot *Display integrated miRNA-mRNA augmented pathways in a dotplot*

Description

This function produces a dotplot that depicts the results of a topologically-aware integrative pathway analysis (TAIPA) carried out through the `topologicalAnalysis()` function.

Usage

```
integrationDotplot(
  object,
  showTerms = 10,
  showTermsParam = "normalized.score",
  title = NULL
)
```

Arguments

object	An object of class IntegrativePathwayAnalysis
showTerms	It is the number of pathways to be shown, based on the order determined by the parameter showTermsParam; or, alternatively, a character vector indicating the pathways to plot. Default is 10
showTermsParam	The order in which the top pathways are selected as specified by the showTerms parameter. It must be one of coverage, padj, pval, score and normalized.score (default)
title	The title of the plot. Default is NULL not to include a plot title

Details

When producing the dotplot with this function, significant pathways are ordered on the x-axis on the basis of their normalized pathway score computed by topologicalAnalysis(). The higher is this score, and the more affected a pathway is between biological conditions. Moreover, the size of each dot is equal to the ratio between the number of nodes for which a measurement is available, and the total number of nodes (pathway coverage). Finally, the color scale of dots is relative to the adjusted p-values of each pathway.

Value

A ggplot graph with a dotplot of integrated pathways.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example IntegrativePathwayAnalysis object
obj <- loadExamples("IntegrativePathwayAnalysis")

# create a dotplot of integrated pathways
integrationDotplot(obj)
```

IntegrativePathwayAnalysis-class
The IntegrativePathwayAnalysis *class*

Description

This class stores the output of integrative multi-omic pathway analyses. In particular, the slots of this class are suitable to represent the results of topologically-aware integrative pathway analysis (TAIPA) returned from the topologicalAnalysis() function.

Usage

```
## S4 method for signature 'IntegrativePathwayAnalysis'
integratedPathways(object)

## S4 method for signature 'IntegrativePathwayAnalysis'
integrationDatabase(object)

## S4 method for signature 'IntegrativePathwayAnalysis'
augmentedPathways(object)

## S4 method for signature 'IntegrativePathwayAnalysis'
show(object)
```

Arguments

object	An object of class IntegrativePathwayAnalysis containing the results of a miRNA-mRNA pathway analysis

Details

Analysis results:

The data slot of this class consists in a data.frame object with six columns, namely:

- pathway, which indicates the name of the biological network;
- coverage, which specifies the fraction of nodes with expression measurement available;
- score, which expresses the score of each individual pathway;
- normalized.score, which indicates the pathway scores after standardizing the values for the null distribution computed through permutations;
- P.Val, the resulting p-value of each pathway;
- adj.P.Val, the p-value adjusted for multiple testing.

Organisms and databases:

The organism and database slots specify the organism in study and the database used for retrieving biological interactions, respectively. In particular, the topologicalAnalysis() function supports KEGG, WikiPathways, and Reactome databases. Regarding organisms, the supportedOrganisms() function can be used to retrieve the available species for each database.

Statistical significance of the permutation test:

`pCutoff` and `pAdjustment` slots refer to the cutoff used for the analysis. `pCutoff` is the threshold used for defining statistically significant pathways, whereas `pAdjustment` refers to the multiple testing correction method used.

Furthermore, since the statistical significance of each pathway is defined on the basis of a permutation test, the number of permutations is also specified in the `nPerm` slot.

Augmented pathways:

The `pathways` slot contains a `list` with weighted `graph` objects, each representing a biological pathway. These networks have been enlarged by adding the observed miRNA-mRNA interactions. Each network has been processed so that the weight of each edge is +1 for activation interactions, and -1 for repression interactions, such as those occurring between miRNAs and mRNAs.

Differential expression results for both miRNAs and genes:

The expression variation of all miRNAs and genes measured in the study is stored in the `expression` slot. In particular, this slot consists of a `data.frame` object with different information, including log2 fold changes, node weights and p-values.

Minimum percentage of measured features:

The `minPc` slot indicates the minimum percentage of miRNAs/mRNAs above which pathways have been considered for the integrative analysis. This is needed because often, when differential expression analysis is performed, lowly expressed features are removed. Therefore, some pathways might result significantly affected even if only 1% of nodes is perturbed.

Value

- the `integratedPathways()` function returns a `data.frame` with the results of the integrative pathway analysis.
- the `integrationDatabase()` function returns a `character` object with the database used for the analysis.
- the `augmentedPathways()` function returns a `list` object containing the miRNA-augmented pathways considered for TAIPA.

Functions

- `integratedPathways(IntegrativePathwayAnalysis)`: Access the results of integrative miRNA-mRNA pathway analysis
- `integrationDatabase(IntegrativePathwayAnalysis)`: View the database used for the integrative pathway analysis
- `augmentedPathways(IntegrativePathwayAnalysis)`: Extract the list of biological networks augmented with miRNA-mRNA interactions
- `show(IntegrativePathwayAnalysis)`: Show method for objects of class IntegrativePathwayAnalysis

Slots

`data` A `data.frame` object that contains the results of the integrative pathway analysis. See the *details* section for further details

`method` The method used for the analysis

`organism` The name of the organism under consideration (e.g. `Homo sapiens`)

`database` The name of the database used for retrieving biological pathways (e.g. `KEGG`)

`pCutoff` A `numeric` value defining the threshold used for statistical significance (e.g. `0.05`)

`pAdjustment` A `character` indicating the method used to correct p-values for multiple testing (e.g. `fdr`)

`pathways` A `list` of `graph` objects containing the biological networks retrieved from `database`, and augmented with miRNA-mRNA interactions

`expression` A `data.frame` object containing differential expression results for both miRNAs and genes

`minPc` The minimum percentage of measured features that a pathway must have for being considered in the analysis

`nPerm` The number of permutation used for assessing the statistical significance of each pathway

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

`listPathways` *List all the available biological pathways in KEGG, Reactome and WikiPathways*

Description

This function can be used to retrieve a list of valid biological pathways present in KEGG, Reactome and WikiPathways.

Usage

```
listPathways(organism, database)
```

Arguments

`organism`	The name of the organism under consideration. The different databases have different supported organisms. To see the list of supported organisms for a given database, use the `supportedOrganisms()` function
`database`	The name of the database to use. It must be one of: `KEGG`, `Reactome`, and `WikiPathways`

Value

A `character` vector containing the pathway names present in the specified database.

Note

This function uses the graphite package to retrieve biological pathways from KEGG, Reactome and WikiPathways.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Sales, G., Calura, E., Cavalieri, D. et al. graphite - a Bioconductor package to convert pathway topology to gene network. BMC Bioinformatics 13, 20 (2012), https://doi.org/10.1186/1471-2105-13-20.

Examples

```
# list the mouse pathways present in WikiPathways
listPathways("Mus musculus", "WikiPathways")
```

---

loadExamples *Load example MIRit objects*

Description

This helper function allows to create a MirnaExperiment object containing miRNA and gene expression data deriving from Riesco-Eizaguirre et al (2015), an IntegrativePathwayAnalysis object containing TAIPA results for the same dataset, or a FunctionalEnrichment with example GSEA enrichment results.

Usage

```
loadExamples(class = "MirnaExperiment")
```

Arguments

class	It must be MirnaExperiment (default) to load an example object of class MirnaExperiment, IntegrativePathwayAnalysis, to load an example object of class IntegrativePathwayAnalysis, or FunctionalEnrichment, to load an example object of class FunctionalEnrichment.

Value

An example MirnaExperiment object, an IntegrativePathwayAnalysis object, or a FunctionalEnrichment object.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# load example IntegrativePathwayAnalysis object
obj <- loadExamples("IntegrativePathwayAnalysis")

# load example FunctionalEnrichment object
obj <- loadExamples("FunctionalEnrichment")
```

`mirnaCounts` *Count matrix for microRNA expression in thyroid cancer*

Description

This dataset contains a gene expression matrix resulting from a miRNA-Seq analysis of thyroid cancer. Specifically, these data originate from Riesco-Eizaguirre et al (2015), where they sequenced 8 papillary thyroid carcinomas (PTC) together with paired samples of normal thyroid tissue. The same thing was done for mRNAs in order to investigate the effects on target genes. Data included in this package have been obtained through the Gene Expression Omnibus (GEO accession: GSE63511).

Usage

```
data(mirnaCounts)
```

Format

`mirnaCounts`**:**

A `matrix` object containing samples as columns and microRNAs as rows.

Value

A `matrix` object with 2576 rows and 16 columns.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE63511

References

Garcilaso Riesco-Eizaguirre et al., "The MiR-146b-3p/PAX8/NIS Regulatory Circuit Modulates the Differentiation Phenotype and Function of Thyroid Cells during Carcinogenesis," Cancer Research 75, no. 19 (September 30, 2015): 4119–30, https://doi.org/10.1158/0008-5472.CAN-14-3547.

MirnaExperiment *The constructor function for MirnaExperiment*

Description

This is the constructor function that allows to easily create objects of class `MirnaExperiment`. This function requires as inputs miRNA and gene expression matrices, as well as sample metadata.

Usage

```
MirnaExperiment(mirnaExpr, geneExpr, samplesMetadata, pairedSamples = TRUE)
```

Arguments

`mirnaExpr`	A `matrix` object containing microRNA expression levels. Other objects coercible to `matrix` are also accepted (e.g. `data.frame`). This object must be structured as specified in the *details* section
`geneExpr`	A `matrix` object containing gene expression levels. Other objects coercible to `matrix` are also accepted (e.g. `data.frame`). This object must be structured as specified in the *details* section
`samplesMetadata`	A `data.frame` object containing information about samples used for microRNA and gene expression profiling. For further information see the *details* section
`pairedSamples`	Logical, whether miRNA and gene expression levels derive from the same subjects or not. Check the *details* section for additional instructions. Default is `TRUE`

Details

This function requires data to be prepared as described below.

mirnaExpr and geneExpr:

`mirnaExpr` and `geneExpr` must be `matrix` objects (or objects coercible to one) that contain miRNA and gene expression values, respectively. Rows must represent the different miRNAs/genes analyzed while columns must represent the different samples in study. For `mirnaExpr`, row names must contain miRNA names according to miRBase nomenclature, whereas for `geneExpr`, row names must contain gene symbols according to hgnc nomenclature. The values contained in these objects can derive from both microarray and RNA-Seq experiments.

For NGS experiments, `mirnaExpr` and `geneExpr` should just be un-normalized count matrices. Instead, for microarray experiments, data should be normalized and log2 transformed, for example with the RMA algorithm.

samplesMetadata:

`samplesMetadata` must be a `data.frame` object containing information about samples used for miRNA profiling and for gene expression analysis. Specifically, this data.frame must contain:

- A column named `primary`, specifying an identifier for each sample;

- A column named `mirnaCol`, containing the column names used for each sample in the `mirnaExpr` object;
- A column named `geneCol`, containing the column names used for each sample in the `geneExpr` object;
- Other eventual columns that define specific sample metadata, such as disease condition, age, sex and so on...

For unpaired samples, NAs can be used for missing entries in `mirnaCol`/`geneCol`.

pairedSamples:

MicroRNA and gene expression measurements may derive from the same subjects (i.e. samples used to generate both miRNA and gene expression data) or from different individuals (i.e. miRNA expression assayed on a group of samples and gene expression retrieved from a different group of samples). `pairedSamples` is a `logical` parameter that defines the relationship between miRNA and gene expression measurements. It must be `TRUE` if data derive from the same individuals, while it must be `FALSE` when data derive from different subjects.

Value

A valid `MirnaExperiment` object containing information about miRNA and gene expression.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example data
data(geneCounts, package = "MIRit")
data(mirnaCounts, package = "MIRit")

# create samples metadata
meta <- data.frame(
    "primary" = colnames(geneCounts),
    "mirnaCol" = colnames(mirnaCounts), "geneCol" = colnames(geneCounts),
    "disease" = c(rep("PTC", 8), rep("NTH", 8)),
    "patient" = c(rep(paste("Sample_", seq(8), sep = ""), 2))
)

# create a 'MirnaExperiment' object
obj <- MirnaExperiment(
    mirnaExpr = mirnaCounts, geneExpr = geneCounts,
    samplesMetadata = meta, pairedSamples = TRUE
)
```

MirnaExperiment-class *The 'MirnaExperiment' class*

Description

This class extends the MultiAssayExperiment from the homonym package to provide the flexibility of handling genomic data of both microRNAs and their targets, allowing to store information about microRNA and gene expression, differential expression results, microRNA targets and miRNA-gene integration analysis.

This class can be used to manage genomic data deriving from different sources, like RNA-Seq, microarrays and mass spectrometry. Moreover, microRNA and gene expression levels may originate from the same individuals (paired samples) or from different subjects (unpaired samples).

Usage

```
## S4 method for signature 'MirnaExperiment'
mirnaDE(object, onlySignificant = TRUE, param = FALSE, returnObject = FALSE)

## S4 method for signature 'MirnaExperiment'
geneDE(object, onlySignificant = TRUE, param = FALSE, returnObject = FALSE)

## S4 method for signature 'MirnaExperiment'
significantMirnas(object)

## S4 method for signature 'MirnaExperiment'
significantGenes(object)

## S4 method for signature 'MirnaExperiment'
pairedSamples(object)

## S4 method for signature 'MirnaExperiment'
mirnaTargets(object)

## S4 method for signature 'MirnaExperiment'
integration(object, param = FALSE)

## S4 method for signature 'MirnaExperiment'
show(object)
```

Arguments

object	An object of class MirnaExperiment
onlySignificant	Logical, if TRUE differential expression results will be returned just for statistically significant miRNAs/genes, if FALSE the full table of miRNA/gene differential expression will be provided. Default is TRUE to only report significant miRNAs/genes

param	Logical, whether to return the complete list object with the parameters used, or just the results stored in data. Default is FALSE
returnObject	Logical, if TRUE this function will return the limma/edgeR/DESeq2 object used for differential expression analysis

Value

- the mirnaDE() function returns a data.frame object with the results of miRNA differential expression analysis.
- the geneDE() function returns a data.frame object with the results of gene differential expression analysis.
- the significantMirnas() function gives back a character vector with the names of differentially expressed miRNAs.
- the significantGenes() function gives back a character vector with the names of differentially expressed genes.
- the pairedSamples() function returns a logical object that specifies if the MirnaExperiment object has paired measurements or not.
- the mirnaTargets() function provides a data.frame object with the list of target genes for the differentially expressed miRNAs.
- the integration() function gives back a data.frame object with the results of the integrative miRNA-mRNA analysis.

Functions

- mirnaDE(MirnaExperiment): Access the results of miRNA differential expression
- geneDE(MirnaExperiment): Access the results of gene differential expression
- significantMirnas(MirnaExperiment): Access the names of differentially expressed miRNAs
- significantGenes(MirnaExperiment): Access the names of differentially expressed genes
- pairedSamples(MirnaExperiment): Check if the object derives from sample-matched data
- mirnaTargets(MirnaExperiment): Extract the miRNA-targets interactions retrieved for the differentially expressed miRNAs
- integration(MirnaExperiment): Access the results of the integrative miRNA-mRNA analysis
- show(MirnaExperiment): Show method for objects of class MirnaExperiment

Slots

`ExperimentList` An `ExperimentList` class object for each assay dataset

`colData` A `DataFrame` of all clinical/specimen data available across experiments

`sampleMap` A `DataFrame` of translatable identifiers of samples and participants

`metadata` Additional data describing the object

`drops` A metadata `list` of dropped information

`mirnaDE` A `list` object containing the results of miRNA differential expression

`geneDE` A `list` object containing the results of gene differential expression

`pairedSamples` A `logical` parameter that specifies whether miRNA and gene expression measurements derive from the same individuals (`TRUE`) or from different subjects (`FALSE`)

`targets` A `data.frame` object containing miRNA-target pairs. This slot is commonly populated by the `getTargets()` function

`integration` A `list` object containing the results of the integration analysis between miRNA and gene expression values. This slot is commonly populated by the `mirnaIntegration()` function

ExperimentList

The `ExperimentList` slot is designed to contain results from each experiment/assay. In this case, it holds miRNA and gene expression matrices. It contains a `SimpleList-class`.

colData

The `colData` slot is a collection of primary specimen data valid across all experiments. This slot is strictly of class `S4Vectors::DataFrame` but arguments for the constructor function allow arguments to be of class `data.frame` and subsequently coerced.

sampleMap

The `sampleMap` contains a `S4Vectors::DataFrame` of translatable identifiers of samples and participants or biological units. The standard column names of the sampleMap are "assay", "primary", and "colname". Note that the "assay" column is a factor corresponding to the names of each experiment in the `ExperimentList`. In the case where these names do not match between the sampleMap and the experiments, the documented experiments in the `sampleMap` take precedence and experiments are dropped by the harmonization procedure. The constructor function will generate a `sampleMap` in the case where it is not provided and this method may be a 'safer' alternative for creating the `MultiAssayExperiment` (so long as the rownames are identical in the `colData`, if provided). An empty `sampleMap` may produce empty experiments if the levels of the "assay" factor in the `sampleMap` do not match the names in the `ExperimentList`.

mirnaDE and geneDE

`mirnaDE` and `geneDE` consist of two `list` objects storing information regarding miRNA and gene differential expression, including:

- `data`, which contains differential expression results in a `data.frame` with five columns:

  - ID: indicates the name of the miRNA/gene;
  - logFC: indicates the fold change of each feature in logarithmic scale;
  - AveExpr: represents the average expression of each miRNA/gene;
  - P.Value: indicates the resulting p-value;
  - adj.P.Val: contains the p-values adjusted for multiple testing.
- significant, which is a character vector containing the names of significantly differentially expressed miRNAs/genes that passed the thresholds;
- method, which specifies the procedure used to determine differentially expressed miRNAs/gens (eg. "limma-voom", "edgeR", "DESeq2", "limma");
- group, which is the column name of the variable (in colData) used for differential expression analysis;
- contrast, which represents the groups that are compared during differential expression analysis (e.g. 'disease-healthy');
- design, which outlines the R formula used for fitting the model. It includes the variable of interest (group) together with eventual covariates (e.g. '~ 0 + disease + sex');
- pCutoff, which indicates the p-value cutoff used for DE analysis;
- pAdjustment, the approach used for multiple testing correction;
- logFC, which states the log2 Fold Change cutoff used for DE analysis;
- deObject, an object deriving from limma/edgeR/DESeq2, that holds additional information regarding data processing.

MiRNA differential expression results can be accessed through the mirnaDE() function, for additional details see ?mirnaDE. Similarly, gene differential expression results can be accessed through the geneDE() function, for additional details see ?geneDE.

pairedSamples

As already mentioned, pairedSamples must be TRUE when miRNA and gene expression derive from the same subjects, while it must FALSE if this is not the case.

targets

targets is a data.frame with miRNA-target interactions, as retrieved by getTargets() function.

integration

Lastly, integration slot contains a list object that stores the results and the options used for performing the integrative miRNA-gene analysis. In particular, integration contains:

- data, which is a data.frame object with the results of the integrative analysis;
- method, which specifies the procedure used to perform the integrative analysis;
- pCutoff, which indicates the p-value cutoff used for the analysis;
- pAdjustment, the approach used for multiple testing correction.

Moreover, data differs on the basis of the integration strategy used. For the one-sided association test integration, and for integration based on rotation gene set tests, this data.frame has seven columns:

- `microRNA`: the miRNA ID;
- `mirna.direction`: the fold change direction of the DE-miRNA (`Up` or `Down`);
- `gene.direction`: the fold change direction of target genes (`Up` or `Down`);
- `DE`: represents the number of differentially expressed targets;
- `targets`: represents the total number of targets for this miRNA;
- `P.Val`: indicates the resulting p-value;
- `adj.P.Val`: contains test p-values corrected for multiple testing;
- `DE.targets`: contains the list of differentially expressed targets whose expression is negatively associated with miRNA expression.

Instead, when a correlation analysis is performed, `data` has six columns:

- `microRNA`: the miRNA ID;
- `Target`: the correlated target gene;
- `microRNA.Direction`: the fold change direction of the DE-miRNA;
- `Corr.Coeff`: the value of the correlation coefficient used;
- `Corr.P.Value`: the p-value resulting from the correlation analysis;
- `Corr.Adjusted.P.Val`: contains the correlation p-values corrected for multiple testing.

To access the results of the integrative analysis, the `data` slot can be accessed through the `integration()` function.

Note

To create a `MirnaExperiment` object, you can use the `MirnaExperiment()` constructor function, which allows to easily build and verify a valid object starting from miRNA and gene expression matrices.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Marcel Ramos et al. Software For The Integration Of Multiomics Experiments In Bioconductor. Cancer Research, 2017 November 1; 77(21); e39-42. DOI: 10.1158/0008-5472.CAN-17-0344

Ronchi, J., & Foti, M. (2026). MIRit: An integrative R framework for the identification of impaired miRNA–mRNA regulatory networks in complex diseases. Bioinformatics Advances, vbag042. https://doi.org/10.1093/bioadv/vbag042

See Also

See `MultiAssayExperiment-class` for additional information.

`mirnaIntegration` *Integrate microRNA and gene expression*

Description

This function allows to identify microRNAs that are significantly associated/correlated with their targets. The principle is that, since the biological role of miRNAs is mainly to negatively regulate gene expression post-transcriptionally, the expression of a microRNA should be negatively correlated with the expression of its targets. To test this assumption for matched-sample data, this function performs a correlation analysis. On the other hand, for unpaired data, it offers different one-sided association tests to estimate if targets of down-regulated miRNAs are enriched in up-regulated genes and vice versa. Additionally, for unpaired data, miRNA effects on target gene expression can also be quantified through a fast approximation to rotation gene-set testing ('fry' method). For correlation analyses, the default behavior is to use Spearman's correlation analysis, whereas for association tests the default option makes use of a one-sided Boschloo's exact test. See the *details* section for further information.

Usage

```
mirnaIntegration(
  mirnaObj,
  test = "auto",
  pCutoff = 0.05,
  pAdjustment = "fdr",
  corMethod = "spearman",
  corCutoff = 0.5,
  partial = FALSE,
  partialCovs = NULL,
  associationMethod = "boschloo",
  nuisanceParam = 100,
  BPPARAM = bpparam()
)
```

Arguments

`mirnaObj`	A `MirnaExperiment` object containing miRNA and gene data
`test`	The statistical test to evaluate the association between miRNAs and genes. It must be one of `auto` (default), to automatically determine the appropriate statistical test; `correlation`, to perform a correlation analysis; `association`, to perform a one-sided association test; `fry` to perform the integrative analysis through rotation gene-set testing
`pCutoff`	The adjusted p-value cutoff to use for statistical significance. The default value is `0.05`. When a lot of interactions are considered, a p-value cutoff after multiple testing correction could result excessively restrictive. In such cases, it is wise to just consider a threshold on the correlation strength and ignore p-values by setting `pCutoff = 1`.

pAdjustment — The p-value correction method for multiple testing. It must be one of: `fdr` (default), `BH`, `none`, `holm`, `hochberg`, `hommel`, `bonferroni`, `BY`

corMethod — The correlation method to be used for correlation analysis. It must be one of: `spearman` (default), `pearson`, `kendall`. See the *details* section for further information

corCutoff — The minimum (negative) value of correlation coefficient to consider meaningful a miRNA-target relationship. Default is `0.5`

partial — Logical, whether a partial correlation analysis should be performed. Default is `FALSE`. See the **details section** for further information

partialCovs — Additional covariates to be considered in partial correlation analysis. This parameter is only considered when `TRUE`. It is an optional parameter that allows to include other covariates in the analysis in addition to `group`

associationMethod — The statistical test used for evaluating the association between miRNAs and their targets for unpaired data. It must be one of `boschloo` (default), to perform a one-sided Boschloo's exact test; `fisher-midp`, to compute a one-sided Fisher's exact test with Lancaster's mid-p correction; `fisher`, to perform a one-sided Fisher's exact test

nuisanceParam — The number of nuisance parameter values considered for p-value calculation in `boschloo` method. The higher this value, the better the p-value estimation accuracy. Default is 100

BPPARAM — The desired parallel computing behavior. This parameter defaults to `BiocParallel::bpparam()`, but this can be edited. See `BiocParallel::bpparam()` for information on parallel computing in R

Details

As already pointed out, if miRNA and gene expression data derive from the same samples, a correlation analysis is used. For evaluating these relationships, the default method used is Spearman's correlation coefficient, as:

- it does not need normally distributed data;
- it does not assume linearity;
- it is much more resistant to outliers.

However, the user can also decide to use other correlation methods, such as Pearson's and Kendall's correlation. Nevertheless, for NGS data it may happen that a certain number of ties is present in the expression values. This can be handled by `spearman` method as it computes a tie-corrected version of Spearman's coefficients. However, another correlation method that is suitable to perform rank correlation on tied data is the Kendall's tau-b method, usable with `kendall`.

Regarding correlation direction, since miRNAs mainly act as negative regulators, only negatively correlated miRNA-target pairs are evaluated, and statistical significance is calculated through a one-tailed t-test.

Additionally, when enough observations are present, it is appropriate to account for the group effect by performing a partial correlation analysis. In particular, a partial correlation analysis evaluates the strength and the direction of a relationship between two variables – miRNA and gene expression in

our case – while accounting for the effect of other factors. In integrative miRNA-mRNA analyses, the group effect considered for differential expression analysis may lead to the identification of several spurious correlated pairs, which result anti-correlated simply because they are dysregulated in opposing directions (upregulated miRNA and downregulated gene). This phenomenon, known as Simpson's paradox, may therefore inflate false positive relationships. By accounting for the group variable using partial correlation analysis, the association between miRNA and gene expression is evaluated within each group, thereby leading to reliable identification of influential miRNAs. To perform such analysis, the `partial` argument must be set to `TRUE`. Furthermore, the effect of other covariates can be considered by passing a `character` vector with the names of variables to account for to the `partialCovs` parameter. However, partial correlation analyses are only effective when a medium-large number of samples are available in each group. Our simulations show that partial correlation outperforms standard correlation when there are at least 20–30 samples for each condition, this is way the default is set to `partial = FALSE`. Furthermore, for batch effects that individually affect either miRNA or gene expression matrices, the only way is to remove them using the `batchCorrection()` function implemented in MIRit.

Moreover, if gene expression data and miRNA expression data derive from different samples (unpaired data), a correlation analysis can't be performed. However, one-sided association tests can be applied in these cases to evaluate if targets of down-regulated miRNAs are statistically enriched in up-regulated genes, and, conversely, if targets of up-regulated miRNAs are statistically enriched in down-regulated genes. In this case, Fisher's exact test can be used to assess the statistical significance of this inverse association. Moreover, Lancaster's mid-p adjustment can be applied since it has been shown that it increases statistical power while retaining Type I error rates. However, Fisher's exact test is a conditional test that requires the sum of both rows and columns of a contingency table to be fixed. Notably, this is not true for genomic data because it is likely that different datasets may lead to a different number of DEGs. Therefore, the default behavior in MIRit is to use a variant of Barnard's exact test, named Boschloo's exact test, that is suitable when group sizes of contingency tables are variable. Moreover, it is possible to demonstrate that Boschloo's test is uniformly more powerful compared to Fisher's exact test.

Finally, for unpaired data, the effect of DE-miRNAs on the expression of target genes can be estimated through rotation gene-set tests. In particular, a fast approximation to rotation gene-set testing called `fry`, implemented in the `limma` package, can be used to statistically quantify the influence of miRNAs on the expression changes of their target genes.

To speed up the identification of anti-correlated or anti-associated miRNA-target pairs, this function implements parallel computation via `BiocParallel::bpparam()`. In this regard, the parallelization behavior can be specified via the `BPPARAM` parameter.

Value

A `MirnaExperiment` object containing integration results. To access these results, the user can make use of the `integration()` function. For additional details on how to interpret the results of miRNA-gene integrative analysis, please see `MirnaExperiment`.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK (2015). "limma powers differential expression analyses for RNA-sequencing and microarray studies." Nucleic Acids Research, 43(7), e47. doi:10.1093/nar/gkv007.

Di Wu and others, ROAST: rotation gene set tests for complex microarray experiments, Bioinformatics, Volume 26, Issue 17, September 2010, Pages 2176–2182, https://doi.org/10.1093/bioinformatics/btq401.

Routledge, R. D. (1994). Practicing Safe Statistics with the Mid-p. The Canadian Journal of Statistics / La Revue Canadienne de Statistique, 22(1), 103–110, https://doi.org/10.2307/3315826.

Boschloo R.D. (1970). "Raised Conditional Level of Significance for the 2x2-table when Testing the Equality of Two Probabilities". Statistica Neerlandica. 24: 1–35. doi:10.1111/j.1467-9574.1970.tb00104.x.

Simpson, E. H. (1951). The Interpretation of Interaction in Contingency Tables. Journal of the Royal Statistical Society: Series B (Methodological), 13(2), 238–241. https://doi.org/10.1111/j.2517-6161.1951.tb00088.x

Ronchi, J., & Foti, M. (2026). MIRit: An integrative R framework for the identification of impaired miRNA–mRNA regulatory networks in complex diseases. Bioinformatics Advances, vbag042. https://doi.org/10.1093/bioadv/vbag042

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# perform integration analysis with default settings
obj <- mirnaIntegration(obj)
```

`mirnaTargets` *Explore miRNA-target pairs*

Description

This function accesses the `targets` slot of a `MirnaExperiment` object. After retrieving miRNA targets with the `getTargets()` function, the interactions between miRNAs and target genes are stored in the `targets` slot and can be explored with this function.

Usage

```
mirnaTargets(object)
```

Arguments

`object`	A `MirnaExperiment` object containing miRNA and gene data

Value

A `data.frame` object containing the interactions between miRNAs and target genes, as retrieved with the `getTargets()` function.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# visualize targets
targets_df <- mirnaTargets(obj)
```

---

mirVariantPlot *Create a trackplot to show the association between miRNAs and disease-SNPs*

Description

This function plots a trackplot that shows the genomic position of disease-associated SNPs that affect miRNA genes. This is useful to visualize the genomic position and context of disease-associated variants that may affect miRNA expression.

Usage

```
mirVariantPlot(
  variantId,
  snpAssociation,
  showContext = FALSE,
  showSequence = TRUE,
  snpFill = "lightblue",
  mirFill = "orange",
  from = NULL,
  to = NULL,
  title = NULL,
  ...
)
```

Arguments

`variantId`	A valid name of a SNP variant! (e.g. `"rs394581"`)
`snpAssociation`	A `data.frame` object containing the results of `findMirnaSNPs()` function

showContext	Logical, if TRUE a complete genomic context with genes present in the region will be shown. Default is FALSE to just display the variant and the miRNA gene
showSequence	Logical, whether to display a color-coded sequence at the bottom of the trackplot. Default is TRUE. This parameter will be set to FALSE if showContext is TRUE
snpFill	It must be an R color name that specifies the fill color of the SNP locus. Default is lightblue. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
mirFill	It must be an R color name that specifies the fill color of the miRNA locus. Default is orange. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
from	The start position of the plotted genomic range. Default is NULL to automatically determine an appropriate position
to	The end position of the plotted genomic range. Default is NULL to automatically determine an appropriate position
title	The title of the plot. Default is NULL not to include a plot title
...	Other parameters that can be passed to Gviz::plotTracks() function

Value

A trackplot with information about chromosome, SNP and miRNA gene location.

Note

This function retrieves genomic coordinates from the output of findMirnaSNPs() function and then uses Gviz package to build the trackplot.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Hahne, F., Ivanek, R. (2016). Visualizing Genomic Data Using Gviz and Bioconductor. In: Mathé, E., Davis, S. (eds) Statistical Genomics. Methods in Molecular Biology, vol 1418. Humana Press, New York, NY. https://doi.org/10.1007/978-1-4939-3578-9_16

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# retrieve associated SNPs
association <- findMirnaSNPs(obj, "response to antidepressant")

# visualize association as a trackplot
mirVariantPlot(variantId = "rs2402960", snpAssociation = association)
```

pairedSamples *View the relationship between miRNA and gene samples*

Description

This function allows to access the pairedSamples slot of a MirnaExperiment object. The MirnaExperiment class is able to contain miRNA and gene expression measurements deriving from the same individuals (paired samples), or from different subjects (unpaired samples).

Usage

```
pairedSamples(object)
```

Arguments

object	A MirnaExperiment object containing miRNA and gene data

Value

A logical value that is either TRUE for paired samples, or FALSE for unpaired samples.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# check if an existing MirnaExperiment object derive from paired samples
pairedSamples(obj)
```

plotCorrelation *Plot correlation between miRNAs and genes within biological groups*

Description

This function creates a scatter plot that shows the correlation between miRNA and gene expression levels. This is useful after correlation analysis performed through the `mirnaIntegration()` function, to graphically visualize the quantitative effect of miRNA dysregulations on target gene expression. Furthermore, this function performs linear/monotonic regression to better represent the relationships between miRNA-target pairs.

Usage

```
plotCorrelation(
  mirnaObj,
  mirna,
  gene,
  condition = NULL,
  showCoeff = TRUE,
  regression = TRUE,
  useRanks = FALSE,
  lineCol = "red",
  lineType = "dashed",
  lineWidth = 0.8,
  pointSize = 3,
  colorScale = NULL,
  fontSize = 12,
  fontFamily = "",
  legend = "top",
  borderWidth = 1,
  allBorders = TRUE,
  grid = TRUE
)
```

Arguments

mirnaObj	A `MirnaExperiment` object containing miRNA and gene data
mirna	The name of the miRNA for which we want to observe the correlation
gene	The name of the gene for which we want to observe the correlation
condition	It must be NULL (default) to plot expression based on the group variable used for differential expression analysis. Alternatively, it must be a character/factor object that specifies group memberships (eg. c("healthy, "healthy", "disease", "disease"))
showCoeff	Logical, whether to show the correlation coeffficient or not. Note that the "R" is used for Pearson's correlation", "rho" for Spearman's correlation, and "tau" for Kendall's correlation. Default is TRUE

regression	Logical, whether to display a linear/monotonic regression line that fits miRNA-gene correlation data. Default is TRUE
useRanks	Logical, whether to represent non-parametric correlation analyses (Spearman's and Kendall's correlations) through rank-transformed data. Note that in this case, linear regression is performed on ranked data instead of monotonic regression. Default is FALSE
lineCol	It must be an R color name that specifies the color of the regression line. Default is red. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
lineType	It specifies the line type used for the regression line. It must be either 'blank', 'solid', 'dashed' (default), 'dotted', 'dotdash', 'longdash' or 'twodash'
lineWidth	The width of the fitted regression line (default is 0.8)
pointSize	The size of points in the correlation plot (default is 3)
colorScale	It must be a named character vector where values correspond to R colors, while names coincide with the groups specified in the condition parameter (eg. c("healthy" = "green", "disease" = "red")). Default is NULL, in order to use the default color scale. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
fontSize	The base size for text elements within the plot. Default is 12
fontFamily	The base family for text elements within the plot
legend	The position of the legend. Allowed values are top, bottom, right, left and none. The default setting is top to show a legend above the plot. If none is specified, the legend will not be included in the graph.
borderWidth	The width of plot borders (default is 1)
allBorders	Logical, whetether to show all panel borders, or just the bottom and left borders. Default is TRUE
grid	Logical, whether to show grid lines or not. Default is TRUE

Details

When non-parametric correlation has been performed with the mirnaIntegration() function, a regression line can be fitted through monotonic regression on expression levels, or through linear regression performed on rank-transformed data. Since, ranks do not correspond to real expression values, the default option is to perform monotonic regression to fit a monotonic curve. To do so, this function makes use of the MonoPoly R package, which implements the algorithm proposed by Murray et al. in 2016.

Value

An object of class ggplot containing the correlation scatter plot.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

K. Murray, S. Müller & B. A. Turlach (2016) Fast and flexible methods for monotone polynomial fitting, Journal of Statistical Computation and Simulation, 86:15, 2946-2966, DOI: 10.1080/00949655.2016.1139582.

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# plot correlation between miR-146b and PAX8 with monotonic regression curve
plotCorrelation(obj, "hsa-miR-146b-5p", "PAX8", condition = "disease")
```

---

plotDE *Represent differentially expressed miRNAs/genes as boxplots, barplots or violinplots*

Description

This function is able to produce boxplots, barplots and violinplots that are useful to visualize miRNA and gene differential expression. The user just has to provide a vector of interesting miRNA/genes that he wants to plot (e.g. "hsa-miR-34a-5p", "hsa-miR-146b-5p", "PAX8"). The chart type can be specified through the graph parameter.

Usage

```
plotDE(
  mirnaObj,
  features,
  condition = NULL,
  graph = "boxplot",
  linear = TRUE,
  showSignificance = TRUE,
  starSig = TRUE,
  pCol = "adj.P.Val",
  sigLabelSize = 7,
  digits = 3,
  nameAsTitle = FALSE,
  colorScale = NULL,
  fontSize = 12,
  fontFamily = "",
  legend = "top",
  borderWidth = 1,
  allBorders = FALSE,
  grid = FALSE
)
```

Arguments

mirnaObj	A MirnaExperiment object containing miRNA and gene data
features	A character vector containing the genes/miRNAs to plot
condition	It must be NULL (default) to plot expression based on the group variable used for differential expression analysis. Alternatively, it must be a character/factor object that specifies group memberships (eg. c("healthy, "healthy", "disease", "disease"))
graph	The type of plot to produce. It must be one of boxplot (default), barplot, violinplot
linear	Logical, whether to plot expression levels in linear scale or in log2 space. Default is TRUE in order to use the linear space
showSignificance	Logical, whether to display statistical significance or not. Default is TRUE
starSig	Logical, whether to represent statistical significance through stars. Default is TRUE, and the significance scale is: * for $p < 0.05$, ** for $p < 0.01$, *** for $p < 0.001$, and **** for $p < 0.0001$. If starSig is set to FALSE, p-values or adjusted p-values will be reported on the plot as numbers
pCol	The statistics used to evaluate comparison significance. It must be one of P.Value, to use unadjusted p-values, and adj.P.Val (default), to use p-values corrected for multiple testing
sigLabelSize	The size for the labels used to show statistical significance. Default is 7, which is well suited for representing p-values as significance stars. However, if starSig is set to FALSE, the user might have to downsize this parameter
digits	The number of digits to show when p-values are reported as numbers (when starSig is FALSE). Default is 3
nameAsTitle	Logical, if set to TRUE, the miRNA/gene name will be added as plot title, and the x-axis and legend will be removed. Note that this option is only considered if features contains just one miRNA/gene. Default is FALSE
colorScale	It must be a named character vector where values correspond to R colors, while names coincide with the groups specified in the condition parameter (eg. c("healthy" = "green", "disease" = "red")). Default is NULL, in order to use the default color scale. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
fontSize	The base size for text elements within the plot. Default is 12
fontFamily	The base family for text elements within the plot
legend	The position of the legend. Allowed values are top, bottom, right, left and none. The default setting is top to show a legend above the plot. If none is specified, the legend will not be included in the graph.
borderWidth	The width of plot borders (default is 1)
allBorders	Logical, whetether to show all panel borders, or just the bottom and left borders. Default is FALSE
grid	Logical, whether to show grid lines or not. Default is FALSE

Value

An object of class ggplot containing the plot.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# produce a boxplot for PAX8 and miR-34a-5p
plotDE(obj, features = c("hsa-miR-34a-5p", "PAX8"))
```

---

plotDimensions *Generate multidimensional scaling (MDS) plots to explore miRNA/gene expression distances*

Description

This function performs multidimensional scaling in order to produce a simple scatterplot that shows miRNA/gene expression variations among samples. In particular, starting from a MirnaExperiment object, this functions allows to visualize both miRNA and gene expression in the multidimensional space. Moreover, it is possible to color samples on the basis of specific variables, and this is extremely useful to assess miRNA/gene expression variations between distinct biological groups.

Usage

```
plotDimensions(
  mirnaObj,
  assay,
  condition = NULL,
  dimensions = c(1, 2),
  labels = FALSE,
  boxedLabel = TRUE,
  pointSize = 3,
  pointAlpha = 1,
  colorScale = NULL,
  title = NULL,
  fontSize = 12,
  fontFamily = "",
  legend = "top",
  borderWidth = 1,
  allBorders = TRUE,
  grid = FALSE,
  ...
)
```

Arguments

mirnaObj	A MirnaExperiment object containing miRNA and gene data
assay	The results to display. It must be either 'microRNA', to plot miRNA expression, or 'genes', to produce MDS plot for genes
condition	It must be the column name of a variable specified in the metadata (colData) of a MirnaExperiment object; or, alternatively, it must be a character/factor object that specifies group memberships (eg. c("healthy, "healthy", "disease", "disease"))
dimensions	It is a numeric vector of length 2 that indicates the two dimensions to represent on the plot. Default is c(1, 2) to plot the two dimensions that account for the highest portion of variability
labels	Logical, whether to display labels or not. Default is FALSE
boxedLabel	Logical, whether to show labels inside a rectangular shape (default) or just as text elements
pointSize	The size of points in the MDS plot (default is 3)
pointAlpha	The transparency of points in the MDS plot (default is 1)
colorScale	It must be a named character vector where values correspond to R colors, while names coincide with the groups specified in the condition parameter (eg. c("healthy" = "green", "disease" = "red")). Default is NULL, in order to use the default color scale. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
title	The title of the plot. Default is NULL not to include a plot title
fontSize	The base size for text elements within the plot. Default is 12
fontFamily	The base family for text elements within the plot
legend	The position of the legend. Allowed values are top, bottom, right, left and none. The default setting is top to show a legend above the plot. If none is specified, the legend will not be included in the graph.
borderWidth	The width of plot borders (default is 1)
allBorders	Logical, whetether to show all panel borders, or just the bottom and left borders. Default is TRUE
grid	Logical, whether to show grid lines or not. Default is FALSE
...	Other parameters that can be passed to limma::plotMDS() function

Value

An object of class ggplot containing the plot.

Note

To perform multidimensional scaling, this function internally uses limma::plotMDS() function provided by limma package.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK (2015). "limma powers differential expression analyses for RNA-sequencing and microarray studies." Nucleic Acids Research, 43(7), e47. doi:10.1093/nar/gkv007.

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# produce MDS plot for genes with condition color
plotDimensions(obj, "genes", condition = "disease")
```

`plotVolcano` *Produce volcano plots to display miRNA/gene differential expression*

Description

This function allows the user to create publication-quality volcano plots to represent the results of miRNA/gene differential expression. In this kind of plots, the x-axis is relative to the log2 fold change between biological conditions, while the y-axis contains the negative base-10 logarithm of the p-value. Note, that even if volcano plots display unadjusted p-values on the y-axis, the cutoff level shown in this plot derive from the adjusted p-value cutoff used for differential expression analysis.

Usage

```
plotVolcano(
  mirnaObj,
  assay,
  labels = NULL,
  boxedLabel = TRUE,
  pointSize = 3,
  pointAlpha = 0.4,
  interceptWidth = 0.6,
  interceptColor = "black",
  interceptType = "dashed",
  colorScale = c("blue", "grey", "red"),
  title = NULL,
  fontSize = 12,
  fontFamily = "",
  legend = "none",
  borderWidth = 1,
  allBorders = TRUE,
  grid = FALSE
)
```

Arguments

mirnaObj	A MirnaExperiment object containing miRNA and gene data
assay	The results to display. It must be either 'microRNA', to plot miRNA differential expression, or 'genes', to show the results for genes
labels	The labels to show on the graph. Default is NULL not to include labels. This parameter can be a character vector containing the IDs of the features that you want to display. Alternatively, this parameter can also be the number of most significant features for which we want to plot labels
boxedLabel	Logical, whether to show labels inside a rectangular shape (default) or just as text elements
pointSize	The size of points in the volcano plot (default is 3)
pointAlpha	The transparency of points in the volcano plot (default is 0.4)
interceptWidth	The width of cutoff intercepts (default is 0.6)
interceptColor	It must be an R color name that specifies the color of cutoff intercepts. Default is black. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
interceptType	It specifies the line type used for cutoff intercepts. It must be either 'blank', 'solid', 'dashed' (default), 'dotted', 'dotdash', 'longdash' or 'twodash'
colorScale	It must be a character vector of length 3 containing valid R color names for downregulated, non significant, and upregulated features, respectively. Default value is c('blue', 'grey', 'red'). Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
title	The title of the plot. Default is NULL not to include a plot title
fontSize	The base size for text elements within the plot. Default is 12
fontFamily	The base family for text elements within the plot
legend	The position of the legend. Allowed values are top, bottom, right, left and none. The default setting is none so that the legend will not be included in the graph.
borderWidth	The width of plot borders (default is 1)
allBorders	Logical, whetether to show all panel borders, or just the bottom and left borders. Default is TRUE
grid	Logical, whether to show grid lines or not. Default is FALSE

Value

An object of class ggplot containing the plot.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# produce a volcano plot for genes
plotVolcano(obj, "genes")
```

`preparePathways` *Prepare miRNA-augmented pathways for integrative miRNA-mRNA pathway analyses*

Description

This function takes influential miRNA-mRNA interactions, identified by the `mirnaIntegration()` function, and adds them to biological pathways retrieved from a pathway database such as `KEGG`, `WikiPathways` and `Reactome`. The pathways returned from this function are needed to perform a topologically-aware integrative pathway analysis (TAIPA) through the `topologicalAnalysis()` function.

Usage

```
preparePathways(
  mirnaObj,
  database = "KEGG",
  organism = "Homo sapiens",
  minPc = 10,
  size = NULL,
  BPPARAM = bpparam()
)
```

Arguments

`mirnaObj`	A `MirnaExperiment` object containing miRNA and gene data
`database`	The name of the database to use. It must be one of: `KEGG`, `Reactome`, and `WikiPathways`. Default is `KEGG`
`organism`	The name of the organism under consideration. The different databases have different supported organisms. To see the list of supported organisms for a given database, use the `supportedOrganisms()` function. Default specie is `Homo sapiens`
`minPc`	The minimum percentage of measured features that a pathway must have for being considered in the analysis. Default is 10. See the *details* section for additional information
`size`	An optional numeric vector of length 2, containing the minimum and maximum number of nodes for each pathway. For example, if `size = c(10, 150)`, all the pathways with less than 10 nodes and those with more than 150 nodes will be discarded. Default is NULL to keep all pathways

BPPARAM The desired parallel computing behavior. This parameter defaults to `BiocParallel::bpparam()`, but this can be edited. See `BiocParallel::bpparam()` for information on parallel computing in R

Details

To create augmented pathways, this function uses the `graphite` R package to download biological networks from the above mentioned databases. Then, each pathway is converted to a `graph` object, and significant miRNA-mRNA interactions are added to the network. Further, edge weights are added according to interaction type.

At this point, biological pathways with few nodes measured are excluded from this analysis. This is required because, during differential expression analysis, lowly expressed features are removed. Therefore, some pathways might result significantly affected even if only 1% of nodes is perturbed. The default behavior is to exclude pathways with less than 10% of representation (`minPc = 10`).

Finally, this function performs a breadth-first search (BFS) algorithm to topologically sort pathway nodes so that each individual node occurs after all its upstream nodes. Nodes within cycles are considered leaf nodes.

Information about pathway coverage, i.e. the percentage of nodes with expression measurments, edge weights, topological sorting order, and the parameters used to create the networks are all stored in the `graphData` slot of each `graphNEL` object.

Value

A `list` object containing the miRNA-augmented pathways as `graphNEL` objects.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Sales, G., Calura, E., Cavalieri, D. et al. graphite - a Bioconductor package to convert pathway topology to gene network. BMC Bioinformatics 13, 20 (2012), https://doi.org/10.1186/1471-2105-13-20.

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# perform integration analysis with default settings
obj <- mirnaIntegration(obj)

# retrieve pathways from KEGG and augment them with miRNA-gene interactions
paths <- preparePathways(obj)

# perform the integrative pathway analysis with 1000 permutations
ipa <- topologicalAnalysis(obj, paths, nPerm = 1000)
```

```
# access the results of pathway analysis
integratedPathways(ipa)

# create a dotplot of integrated pathways
integrationDotplot(ipa)

# explore a specific biological network
visualizeNetwork(ipa, "Thyroid hormone synthesis")
```

`searchDisease` *Search for disease EFO identifiers*

Description

This function allows to retrieve the Experimental Factor Ontology (EFO) identifier of a particular disease. This ID is then needed to use the function `findMirnaSNPs()`.

Usage

```
searchDisease(diseaseName)
```

Arguments

`diseaseName`	The name of a particular disease (ex. `Alzheimer disease`).

Value

A `character` object containing EFO identifiers.

Note

To retrieve EFO IDs for specific diseases, this function makes use of the `gwasrapidd` package.

Author(s)

Jacopo Ronchi, `<jacopo.ronchi@unimib.it>`

References

Ramiro Magno, Ana-Teresa Maia, gwasrapidd: an R package to query, download and wrangle GWAS catalog data, Bioinformatics, Volume 36, Issue 2, January 2020, Pages 649–650, https://doi.org/10.1093/bioinformatics/btz605.

Examples

```
# search the EFO identifier of Alzheimer disease
searchDisease("Alzheimer disease")
```

setTargets *Use custom miRNA-target interactions*

Description

Apart from the getTargets() function, users can use their own custom list of miRNA-target interactions. This function allows to directly add custom interactions to an existing MirnaExperiment object.

Usage

```
setTargets(mirnaObj, tg)
```

Arguments

mirnaObj	A MirnaExperiment object containing miRNA and gene data
tg	A data.frame object specifying miRNA-target interactions. Check the **details section** for the format

Details

This function requires interactions to be passed as a data.frame object with two column, one for miRNAs and one for genes. The column names of this data.frame must include MicroRNA and Gene.Symbol.

Value

A MirnaExperiment object containing miRNA targets stored in the targets slot. Results can be accessed with the mirnaTargets() function.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Ronchi, J., & Foti, M. (2026). MIRit: An integrative R framework for the identification of impaired miRNA–mRNA regulatory networks in complex diseases. Bioinformatics Advances, vbag042. https://doi.org/10.1093/bioadv/vbag042

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# define targets
tg <- data.frame(MicroRNA = c("hsa-miR-1179", "hsa-miR-1179"),
                 Gene.Symbol = c("ACVR2A", "ABI2"))
```

```
# set targets
obj <- setTargets(obj, tg)

# print targets
mirnaTargets(obj)
```

`significantAccessors` *Get the IDs of statistically differentially expressed miRNAs/genes*

Description

The `significantMirnas()` and `significantGenes()` functions access the `significant` features contained in the `mirnaDE` or `geneDE` slots of a `MirnaExperiment` object, and can be used to obtain the IDs of statistically differentially expressed miRNAs and genes.

Usage

```
significantMirnas(object)

significantGenes(object)
```

Arguments

`object`	A `MirnaExperiment` object containing miRNA and gene data

Value

A `character` vector of miRNA IDs (e.g. 'hsa-miR-16-5p', hsa-miR-29a-3p'...), or a`character` vector of gene symbols (e.g. 'TP53', 'FOXP2', 'TIGAR', CASP1'...).

Functions

- `significantMirnas()`: Get the IDs of differentially expressed miRNAs
- `significantGenes()`: Get the IDs of differentially expressed genes

Author(s)

Jacopo Ronchi, `<jacopo.ronchi@unimib.it>`

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# extract significant DE-miRNAs
sigMirnas <- significantMirnas(obj)
```

```
# extract significant DEGs
sigGenes <- significantGenes(obj)
```

`supportedOrganisms` *Get the list of supported organisms for a given database*

Description

This function provides the list of supported organisms for different databases, namely Gene Ontology (GO), Kyoto Encyclopedia of Genes and Genomes (KEGG), MsigDB, WikiPathways, Reactome, Enrichr, Disease Ontology (DO), Network of Cancer Genes (NCG), DisGeNET, and COVID19.

Usage

```
supportedOrganisms(database)
```

Arguments

`database`	The database name. It must be one of: `GO`, `KEGG`, `MsigDB`, `WikiPathways`, `Reactome`, `Enrichr`, `DO`, `NCG`, `DisGeNET`, `COVID19`

Value

A `character` vector listing all the supported organisms for the database specified by the user.

Note

To perform the functional enrichment of genes, MIRit uses the `geneset` R package to download gene sets from the above mentioned databases.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Liu, Y., Li, G. Empowering biologists to decode omics data: the Genekitr R package and web server. BMC Bioinformatics 24, 214 (2023). https://doi.org/10.1186/s12859-023-05342-9.

Examples

```
# get the supported organisms for GO database
supportedOrganisms("GO")

# get the supported organisms for Reactome
supportedOrganisms("Reactome")
```

`topologicalAnalysis` *Perform a topologically-aware integrative pathway analysis (TAIPA)*

Description

This function allows to perform an integrative pathway analysis that aims to identify the biological networks that are most affected by miRNomic and transcriptomic dysregulations. This function takes miRNA-augmented pathways, created by the `preparePathways()` function, and then calculates a score that estimates the degree of impairment for each pathway. Later, statistical significance is calculated through a permutation test. The main advantages of this method are that it doesn't require matched samples, and that it allows to perform an integrative miRNA-mRNA pathway analysis that takes into account the topology of biological networks. See the *details* section for additional information.

Usage

```
topologicalAnalysis(
  mirnaObj,
  pathways,
  pCutoff = 0.05,
  pAdjustment = "max-T",
  nPerm = 10000,
  progress = FALSE,
  tasks = 0,
  BPPARAM = bpparam()
)
```

Arguments

`mirnaObj`	A `MirnaExperiment` object containing miRNA and gene data
`pathways`	A `list` of miRNA-augmented pathways returned by the `preparePathways()` function
`pCutoff`	The adjusted p-value cutoff to use for statistical significance. The default value is `0.05`
`pAdjustment`	The p-value correction method for multiple testing. It must be one of: `max-T` (default), `fdr`, `BH`, `none`, `holm`, `hochberg`, `hommel`, `bonferroni`, `BY`
`nPerm`	The number of permutation used for assessing the statistical significance of each pathway. Default is 10000. See the *details* section for additional information
`progress`	Logical, whether to show a progress bar during p-value calculation or not. Default is FALSE, not to include a progress bar. Please note that setting `progress = TRUE` with high values of `tasks` leads to less efficient parallelization. See the *details* section for additional information
`tasks`	An integer between 0 and 100 that specifies how frequently the progress bar must be updated. Default is 0 to simply split the computation among the workers. High values of `tasks` can lead to 15-30% slower p-value calculation. See the *details* section for additional information

BPPARAM The desired parallel computing behavior. This parameter defaults to `BiocParallel::bpparam()`, but this can be edited. See `BiocParallel::bpparam()` for information on parallel computing in R

Details

Topologically-Aware Integrative Pathway Analysis (TAIPA):

This analysis aims to identify the biological pathways that result affected by miRNA and mRNA dysregulations. In this analysis, biological pathways are retrieved from a pathway database such as KEGG, and the interplay between miRNAs and genes is then added to the networks. Each network is defined as a graph $G(V, E)$, where V represents nodes, and E represents the relationships between nodes.

Then, nodes that are not significantly differentially expressed are assigned a weight $w_i = 1$, whereas differentially expressed nodes are assigned a weight $w_i = |\Delta E_i|$, where ΔE_i is the linear fold change of the node. Moreover, to consider the biological interaction between two nodes, namely i and j, we define an interaction parameter $\beta_{i\to j} = 1$ for activation interactions and $\beta_{i\to j} = -1$ for repression interactions. Subsequently, the concordance coefficient $\gamma_{i\to j}$ is defined as:

$$\gamma_{i\to j} = \begin{cases} \beta_{i\to j} & \text{if } sign(\Delta E_i) = sign(\Delta E_j) \\ -\beta_{i\to j} & \text{if } sign(\Delta E_i) \neq sign(\Delta E_j) \end{cases} .$$

Later in the process, a breadth-first search (BFS) algorithm is applied to topologically sort pathway nodes so that each individual node occurs after all its upstream nodes. Nodes within cycles are considered leaf nodes. At this point, a node score ϕ is calculated for each pathway node i as:

$$\phi_i = w_i + \sum_{j=1}^{U} \gamma_{i\to j} \cdot k_j \, .$$

where U represents the number of upstream nodes, $\gamma_{i\to j}$ denotes the concordance coefficient, and k_j is a propagation factor defined as:

$$k_j = \begin{cases} w_j & \text{if } \phi_j = 0 \\ \phi_j & \text{if } \phi_j \neq 0 \end{cases} .$$

Finally, the pathway score Ψ is calculated as:

$$\Psi = \frac{1 - M}{N} \cdot \sum_{i=1}^{N} \phi_i \, ,$$

where M represents the proportion of miRNAs in the pathway, and N represents the total number of nodes in the network.

Then, to compute the statistical significance of each pathway score, a permutation procedure is applied. Later, both observed pathway scores and permuted scores are standardized by subtracting the mean score of the permuted sets μ_{Ψ_P} and then dividing by the standard deviation of the permuted scores σ_{Ψ_P}.

Finally, the p-value is defined based on the fraction of permutations that reported a higher normalized pathway score than the observed one. However, to prevent p-values equal to zero, we define p-values as:

$$p = \frac{\sum_{n=1}^{N_p} [\Psi_{P_N} \geq \Psi_N] + 1}{N_p + 1} .$$

In the end, p-values are corrected for multiple testing either through the max-T procedure (default option) which is particularly suited for permutation tests, or through the standard multiple testing approaches.

Implementation details:

For computational efficiency, pathway score computation has been implemented in C++ language. Moreover, to define the statistical significance of each network, a permutation test is applied following the number of permutations specified with `nPerm`. The default setting is to perform 10000 permutations. The higher is the number of permutations, the more stable are the calculated p-values, even though the time needed will increase. In this regard, since computing pathway score for 10000 networks for each pathway is computationally intensive, parallel computing has been employed to reduce running time. The user can modify the parallel computing behavior by specifying the `BPPARAM` parameter. See `BiocParallel::bpparam()` for further details. Further, a progress bar can also be included to show the completion percentage by setting `progress = TRUE`. Moreover, the user can define how frequently the progress bar gets updated by tweaking the `tasks` parameter. When using `progress = TRUE`, setting `tasks` to 100 tells the function to update the progress bar 100 times, so that the user can see increases of 1%. Instead, setting `tasks` to 50, means that the progress bar gets updated every 2% of completion. However, keep in mind that `tasks` values from 50 to 100 lead to 15-30% slower p-value calculation due to increased data transfer to the workers. Instead, lower `tasks` values like 20 determine less frequent progress updates but are only slightly less efficient than not including a progress bar.

Value

An object of class `IntegrativePathwayAnalysis` that stores the results of the analysis. See the relative help page for further details.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

References

Peter H. Westfall and S. Stanley Young. Resampling-Based Multiple Testing: Examples and Methods for p-Value Adjustment. John Wiley & Sons. ISBN 978-0-471-55761-6.

Examples

```
# load example MirnaExperiment object
obj <- loadExamples()

# perform integration analysis with default settings
obj <- mirnaIntegration(obj)

# retrieve pathways from KEGG and augment them with miRNA-gene interactions
paths <- preparePathways(obj)
```

```
# perform the integrative pathway analysis with 1000 permutations
ipa <- topologicalAnalysis(obj, paths, nPerm = 1000)

# access the results of pathway analysis
integratedPathways(ipa)

# create a dotplot of integrated pathways
integrationDotplot(ipa)

# explore a specific biological network
visualizeNetwork(ipa, "Thyroid hormone synthesis")
```

`visualizeNetwork` *Visualize the relationships between miRNAs and genes in a biological pathway*

Description

This function can be used to plot augmented pathways created by the `topologicalAnalysis()` function. In particular, given a valid object of class `IntegrativePathwayAnalysis`, this function allows to produce a network graph for a specified biological pathway, alongside with expression fold changes. In this way, augmented pathways made of both miRNAs and genes can be visually explored to better investigate the consequences of miRNA/gene dysregulations.

Usage

```
visualizeNetwork(
  object,
  pathway,
  algorithm = "dot",
  fontsize = 14,
  lfcScale = c("royalblue", "white", "red"),
  nodeBorderCol = "black",
  nodeTextCol = "black",
  edgeCol = "darkgrey",
  edgeWidth = 1,
  subgraph = NULL,
  highlightNodes = NULL,
  highlightCol = "gold",
  highlightWidth = 2,
  legendColorbar = TRUE,
  legendInteraction = TRUE,
  title = NULL,
  titleCex = 2,
  titleFace = 1
)
```

Arguments

`object`	An object of class `IntegrativePathwayAnalysis` containing the results of a miRNA-mRNA pathway analysis
`pathway`	The name of the biological pathway to show. The available pathways for a given database can be seen through the `listPathways()` function
`algorithm`	The layout algorithm used to arrange nodes in the network. It must be one of `dot` (default), `circo`, `fdp`, `neato`, `osage` or `twopi`. For more information regarding these algorithms, please check the *details* section
`fontsize`	The font size of each node in the graph. Default is 14
`lfcScale`	It must be a `character` vector of length 3 containing valid R color names for creating a gradient of log2 fold changes. The first value refers to downregulation, the middle one to stable expression, and the last one to upregulation. Default value is `c('royalblue', 'white', 'red')`. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
`nodeBorderCol`	It must be an R color name that specifies the color of node borders. Default is `black`. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
`nodeTextCol`	It must be an R color name that specifies the color of miRNA/gene names. Default is `black`. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
`edgeCol`	It must be an R color name that specifies the color of edges between nodes. Default is `darkgrey`. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
`edgeWidth`	The width of edges. Default is 1
`subgraph`	An optional `character` vector containing the nodes that you want to maintain in the final plot. All the other nodes will not be shown. This is useful to display specific features of extremely messy graphs. Default is NULL
`highlightNodes`	A `character` vector containing the names of nodes that you want to highlight. Default is NULL not to highlight any nodes. See the *details* section for additional information
`highlightCol`	It must be an R color name that specifies the color of edges and borders for highlighted nodes. Default is `gold`. Available color formats include color names, such as 'blue' and 'red', and hexadecimal colors specified as #RRGGBB
`highlightWidth`	The width of edges between highlighted nodes. Default is 2
`legendColorbar`	Logical, whether to add a legend with a color bar for log2 fold changes. Default is TRUE
`legendInteraction`	Logical, whether to add a legend that links edge types to biological interactions. Default is TRUE
`title`	The title of the plot. Default is `NULL` not to include a plot title
`titleCex`	The cex of the plot main title. Default is 2
`titleFace`	An integer which specifies which font to use for title. 1 corresponds to plain text, 2 to bold face, 3 to italic, 4 to bold italic, and 5 to symbol font. Default is 1

Details

The network created by this function is highly flexible, allowing to tweak different parameters that can influence the resulting graph, including node highlighting, layout algorithms, colors, and legends.

Nodes included in the plot:

For huge messy networks, the user can specify the nodes to include in the plot through the `subgraph` parameter, in order to represent only the features that he wants to display. Alternatively, this parameter can be set to NULL (default), to plot all nodes of that biological pathway.

Highlight nodes and edges:

One interesting feature offered by this function consists in highlighting specific nodes and edges within a network. This results particularly useful when we want to put in evidence affected routes in a biological pathway. To highlight nodes, you must provide the `highlightNodes` parameter with a `character` vector that lists all the desired nodes. As a result, the borders of highlighted nodes will be colored according to `highlightCol` parameter (default is 'gold'), and will have a width specified by `highlightWidth` (default is 2). Notably, this function automatically highlights in the same way the edges connecting the selected nodes.

Layout algorithms:

Furthermore, this function allows to use different methods to lay out nodes in the network by setting the `algorithm` parameter. In this regard, several algorithms in `Rgraphviz` package can be used, namely:

- `dot` (default), which is an algorithm attributed to Sugiyama et al. and described by Gansner et al., that creates a ranked layout that is particularly suited to display hierarchies and complex pathways;
- `circo`, which uses a recursive radial algorithm resulting in a circular layout;
- `fdp`, which adopts a force-directed approach similar to that of Fruchterman and Reingold;
- `neato`, which relies on a spring model where an iterative solver finds low energy configurations;
- `osage`, which is a layout engine that recursively draws cluster subgraphs;
- `twopi`, which places a node in the center of the network, and then arranges the remaining nodes in a series of concentric circles around the center.

For additional information on these algorithms, refer to Rgraphviz::GraphvizLayouts.

Customization:

To customize the look of the resulting plot, this function allows to change different graphical parameters, including:

- the color scale for log2 fold changes, that can be set with `lfcScale`;
- the font size of nodes, which can be changed through `fontsize`;
- the border color for nodes, which can be edited with `nodeBorderCol`;
- the text color of nodes, which can be changed through `nodeTextCol`;
- the color used for edges, set by `edgeCol`;
- the width of edges, customizable with `edgeWidth`.

Additionally, this function allows to include handy legends that are useful for interpreting the biological consequences of network alterations. In particular:

- a color bar legend displaying the log2 fold changes corresponding to each fill color can be included with `legendColorbar = TRUE` (default); and
- a legend that links the appearance of edges and arrow heads to the type of biological interaction can be shown through `legendInteraction = TRUE` (default).

Lastly, `title`, `titleCex` and `titleFace` parameters can be tweaked to include a network title with the desired look.

Value

A base R plot with the augmented pathway.

Note

This function uses the `Rgraphviz` package to render the network object.

Author(s)

Jacopo Ronchi, <jacopo.ronchi@unimib.it>

Examples

```
# load example IntegrativePathwayAnalysis object
obj <- loadExamples("IntegrativePathwayAnalysis")

# explore a specific biological network
visualizeNetwork(obj, "Thyroid hormone synthesis")
```

Index

∗ **datasets**
    geneCounts, 31
    mirnaCounts, 46

addDifferentialExpression, 3
augmentedPathways, 6
augmentedPathways,IntegrativePathwayAnalysis-method *(*IntegrativePathwayAnalysis-class*)*, 42

batchCorrection, 7
batchCorrection(), *56*
BiocParallel::bpparam(), *55, 56*, *70*, *76, 77*

deAccessors, 9
deAnalysis, 10
DESeq2::DESeq(), *12*

edgeR::calcNormFactors(), *12*
edgeR::estimateDisp(), *12*
edgeR::filterByExpr(), *12*
edgeR::glmQLFit(), *12*
edgeR::glmQLFTest(), *12*
enrichedFeatures, 15
enrichedFeatures,FunctionalEnrichment-method *(*FunctionalEnrichment-class*)*, 29
enrichGenes, 16
enrichGenes(), *20–23*, *25, 26*, *29*, *36, 37*
enrichmentBarplot, 20
enrichmentBarplot(), *19*, *26*
enrichmentDatabase, 21
enrichmentDatabase,FunctionalEnrichment-method *(*FunctionalEnrichment-class*)*, 29
enrichmentDotplot, 22
enrichmentDotplot(), *19*, *26*
enrichmentMethod, 23
enrichmentMethod,FunctionalEnrichment-method *(*FunctionalEnrichment-class*)*, 29
enrichmentMetric, 24
enrichmentMetric,FunctionalEnrichment-method *(*FunctionalEnrichment-class*)*, 29
enrichmentResults, 24
enrichmentResults(), *19*, *26*
enrichmentResults,FunctionalEnrichment-method *(*FunctionalEnrichment-class*)*, 29
enrichTargets, 25
ExperimentList, *51*

findMirnaSNPs, 27
findMirnaSNPs(), *58, 59*, *71*
FunctionalEnrichment, *6*, *15*, *19–26*, *29*, *31, 32*, *36–38*, *40*, *45*
FunctionalEnrichment *(*FunctionalEnrichment-class*)*, 29
FunctionalEnrichment-class, 29

geneCounts, 31
geneDE *(*deAccessors*)*, 9
geneDE(), *5*, *14*, *52*
geneDE,MirnaExperiment-method *(*MirnaExperiment-class*)*, 49
geneSet, 31
geneSet,FunctionalEnrichment-method *(*FunctionalEnrichment-class*)*, 29
getEvidence, 32
getTargets, 33
getTargets(), *51, 52*, *57, 58*, *72*
ggplot2::ggplot2, *36*
gseaPlot, 36
gseaPlot(), *19*
gseaRidgeplot, 37

gseaRidgeplot(), *19*
Gviz::plotTracks(), *59*

integratedPathways, 38
integratedPathways,IntegrativePathwayAnalysis-method
(IntegrativePathwayAnalysis-class), 42
integration, 39
integration(), *53*, *56*
integration,MirnaExperiment-method
(MirnaExperiment-class), 49
integrationDatabase, 40
integrationDatabase,IntegrativePathwayAnalysis-method
(IntegrativePathwayAnalysis-class), 42
integrationDotplot, 40
IntegrativePathwayAnalysis, *6*, *38*, *40–42*, *45*, *77–79*
IntegrativePathwayAnalysis
(IntegrativePathwayAnalysis-class), 42
IntegrativePathwayAnalysis-class, 42

limma::arrayWeights(), *13*
limma::duplicateCorrelation(), *13*
limma::eBayes(), *13*
limma::lmFit(), *13*
limma::plotMDS(), *66*
limma::removeBatchEffect(), *8*
limma::voom(), *12*, *13*
limma::voomWithQualityWeights(), *12*, *13*
limma::wsva(), *8*, *13*
listPathways, 44
listPathways(), *79*
loadExamples, 45

mirnaCounts, 46
mirnaDE (deAccessors), 9
mirnaDE(), *5*, *14*, *52*
mirnaDE,MirnaExperiment-method
(MirnaExperiment-class), 49
MirnaExperiment, *3–5*, *7–9*, *12–14*, *16*, *26*, *27*, *34*, *35*, *39*, *45*, 47, *47–49*, *53*, *54*, *56*, *57*, *60*, *61*, *64–66*, *68*, *69*, *72*, *73*, *75*
MirnaExperiment(), *53*
MirnaExperiment-class, 49
mirnaIntegration, 54
mirnaIntegration(), *8*, *39*, *51*, *61*, *62*, *69*
mirnaTargets, 57
mirnaTargets(), *35*, *72*
mirnaTargets,MirnaExperiment-method
(MirnaExperiment-class), 49
mirVariantPlot, 58
mirVariantPlot(), *27*
MultiAssayExperiment, *49*

pairedSamples, 60
pairedSamples,MirnaExperiment-method
(MirnaExperiment-class), 49
performGeneDE (deAnalysis), 10
performGeneDE(), *3*
performMirnaDE (deAnalysis), 10
performMirnaDE(), *3*, *4*
plotCorrelation, 61
plotDE, 63
plotDimensions, 65
plotVolcano, 67
preparePathways, 69
preparePathways(), *75*

Rgraphviz::GraphvizLayouts, *80*

S4Vectors::DataFrame, *51*
searchDisease, 71
searchDisease(), *27*, *32*
setTargets, 72
show,FunctionalEnrichment-method
(FunctionalEnrichment-class), 29
show,IntegrativePathwayAnalysis-method
(IntegrativePathwayAnalysis-class), 42
show,MirnaExperiment-method
(MirnaExperiment-class), 49
significantAccessors, 73
significantGenes
(significantAccessors), 73
significantGenes,MirnaExperiment-method
(MirnaExperiment-class), 49
significantMirnas
(significantAccessors), 73
significantMirnas,MirnaExperiment-method
(MirnaExperiment-class), 49
supportedOrganisms, 74
supportedOrganisms(), *16*, *19*, *26*, *42*, *44*, *69*

topologicalAnalysis, 75

topologicalAnalysis(), *6*, *38*, *40–42*, *69*, *78*

visualizeNetwork, 78