Package 'sSNAPPY'

May 2, 2024

Title Single Sample directioNAl Pathway Perturbation analYsis

Version 1.8.0

Description A single sample pathway perturbation testing method for RNA-seq data. The method propagates changes in gene expression down gene-set topologies to compute single-sample directional pathway perturbation scores that reflect potential direction of change. Perturbation scores can be used to test significance of pathway perturbation at both individual-sample and treatment levels.

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Suggests BiocManager, BiocStyle, colorspace, cowplot, DT, htmltools, knitr, pander, patchwork, rmarkdown, spelling, testthat (>= 3.0.0), tidyverse

Config/testthat/edition 3

SystemRequirements C++11

LazyData false

Imports dplyr (>= 1.1), magrittr, rlang, stats, graphite, tibble, ggraph, igraph, reshape2, org.Hs.eg.db, SummarizedExperiment, edgeR, methods, ggforce, pheatmap, utils, stringr, gtools, tidyr

Depends R (>= 4.3.0), ggplot2

biocViews Software, GeneExpression, GeneSetEnrichment, GeneSignaling

URL https://wenjun-liu.github.io/sSNAPPY/

VignetteBuilder knitr

BugReports https://github.com/Wenjun-Liu/sSNAPPY/issues

Language en-US

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

git_branch RELEASE_3_19

git_last_commit 684bb7e

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-01

Author Wenjun Liu [aut, cre] (<https://orcid.org/0000-0002-8185-3069>),
Stephen Pederson [aut] (<https://orcid.org/0000-0001-8197-3303>)

Maintainer Wenjun Liu <wenjun.liu@adelaide.edu.au>

Contents

`.compute_ssFC` *Compute single sample logFCs*

Description

Compute single sample logFCs

Usage

```
.compute_ssFC(logCPM, metadata, sampleColumn, treatColumn, groupBy)
```

Arguments

`logCPM`	Matrix of normaslised logCPM
`metadata`	Sample metadata data frame as described in the details section.
`factor`	The factor defines how samples can be put into matching pairs.
`control`	The treatment level that is the control.

Value

A matrix of single sample logFC

`generate_permuted_scores` *Permute sample labels to simulate null distribution of perturbation scores*

Description

Simulate null distributions of perturbation scores for each pathway through sample permutation.

Usage

```
generate_permuted_scores(
  expreMatrix,
  NB = NULL,
  testScore = NULL,
  gsTopology,
  weight,
  drop = TRUE
)

## S4 method for signature 'matrix'
generate_permuted_scores(
  expreMatrix,
  NB = NULL,
  testScore = NULL,
  gsTopology,
  weight,
  drop = TRUE
)

## S4 method for signature 'data.frame'
generate_permuted_scores(
  expreMatrix,
  NB = NULL,
  testScore = NULL,
  gsTopology,
  weight,
  drop = TRUE
)

## S4 method for signature 'DGEList'
generate_permuted_scores(
  expreMatrix,
  NB = NULL,
```

```
  testScore = NULL,
  gsTopology,
  weight,
  drop = TRUE
)

## S4 method for signature 'SummarizedExperiment'
generate_permuted_scores(
  expreMatrix,
  NB = NULL,
  testScore = NULL,
  gsTopology,
  weight,
  drop = TRUE
)
```

Arguments

expreMatrix	matrix and data.frame of logCPM, or DGEList/SummarizedExperiment storing gene expression counts. Feature names need to be in entrez IDs
NB	Number of permuted sample pairs to compute permuted logFCs for. Default to be the maximum number of possibilities (See details).
testScore	Optional. Output of pathway_pert() for restricting the permutation only to pathways with non-zero test scores in at least one sample.
gsTopology	List of pathway topology matrices generated using function retrieve_topology
weight	A vector of gene-wise weights derived from function weight_ss_fc
drop	logic(1). Whether to drop pathways with all zero scores

Details

This generate_permuted_scores function firstly randomly permute sample labels to form permuted pairs and generate permuted logFCs, which are then used to compute permuted perturbation scores for each pathway.

The function outputs a list that is of the same length as the list storing pathway topology matrices (i.e. gsTopology). Each element of the output list corresponds to one pathway and contains a vector of permuted perturbation scores. The permuted perturbation scores will be used to approximate the null distributions of perturbation scores and compute permuted p-values.

If the input is S4 object of DGEList or SummarizedExperiment, gene expression matrix will be extracted and converted to a logCPM matrix.

The sample permutation is operated by randomly choosing combination of 2 samples from the column names of the expreMatrix. Hence, sample metadata is not a required input. The number of maximum unique permutations in a dataset with N samples is N x (N-1). By default, permuted logFCs will be computed for all N x (N-1) permuted pairs. However, this can be overwritten by the NB parameter. If the NB specified is smaller than N x (N-1), NB possibilities will be randomly sampled from all the possible permutation. If the NB specified is larger than the maximum number of permutation possible, the parameter will be ignored. Since the smallest achievable permutation

p-value is 1/NB, accurate estimation of small p-value requires a large number of permutations that is only feasible in data with a large sample size.

Value

A list where each element is a vector of perturbation scores for a pathway.

Examples

```
#compute weighted single sample logFCs
data(metadata_example)
data(logCPM_example)
metadata_example <- dplyr::mutate(metadata_example, treatment = factor(
   treatment, levels = c("Vehicle", "E2+R5020", "R5020")))
ls <- weight_ss_fc(logCPM_example, metadata = metadata_example,
 groupBy = "patient", treatColumn = "treatment", sampleColumn = "sample")
## Not run:
load(system.file("extdata", "gsTopology.rda", package = "sSNAPPY"))

# simulate the null distribution of scores through sample permutation
permutedScore <- generate_permuted_scores(logCPM_example,
gsTopology = gsTopology, weight = ls$weight)


## End(Not run)
```

---

gsAnnotation_df *gsAnnotation_df: Categorization of KEGG pathways used for community annotation*

Description

gsAnnotation_df: Categorization of KEGG pathways used for community annotation

Usage

```
data(gsAnnotation_df)
```

Format

A data.frame with 549 rows and 2 columns containing categorization of 549 KEGG pathways

gs_name Gene-set name

vategory Category

Source

https://www.genome.jp/kegg/

`logCPM_example` *logCPM_example: Normalised logCPM of patient-derived explant models obtained from 5 ER-positive primamry breast cancer tumours (GSE80098)*

Description

This data was adopted from a study by Singhal H, et al., which was published as *Genomic agonism and phenotypic antagonism between estrogen and progesterone receptors in breast cancer* in 2016.

Usage

```
data(logCPM_example)
```

Format

A `data.frame` with 7672 rows and 15 columns

Details

In this study, 12 primary malignant breast tissues (8PR+ and 4 PR-) were developed into patient-derived explants and treated with Vehicle, E2, E2+R5020, or R5020 for 24 or 48 hrs.

Raw data for 48-hr Vehicle-, R5020-treated and E2+R5020-treated samples were retrieved from GEO (GSE80098) and pre-processed into raw count. Filtration was sequentially performed to remove undetectable genes and the filtered counts were normalised using conditional quantile normalisation to offset effects of systematic artefacts, such as gene length and GC contents.

To reduce computing time, we randomly sampled half of the genes after filtration and used their logCPM value as the example data.

Source

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE80098

`metadata_example` *metadata_example: Sample metadata for malignant breast cancer tumours PDE from 5 ER+ breast cancer tumour (GSE80098)*

Description

metadata_example: Sample metadata for malignant breast cancer tumours PDE from 5 ER+ breast cancer tumour (GSE80098)

Usage

```
data(metadata_example)
```

Format

A data.frame with 15 rows and 4 columns

patient patient N2-3, P4-6

treatment treatment: Vehicle, R5020 or E2+R5020

PR progesterone receptor status

sample sample name, corresponding to column names of the logCPM matrix

Source

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4928895/

---

normalise_by_permu *Normalise test perturbation scores by permutation result and compute permutation p-values*

Description

Normalise test perturbation scores by permutation result and compute permutation p-values

Usage

```
normalise_by_permu(
  permutedScore,
  testScore,
  pAdj_method = "fdr",
  sortBy = c("gs_name", "sample", "pvalue")
)
```

Arguments

permutedScore	A list. Output of generate_permuted_scores
testScore	A data.frame. Output of pathway_pert
pAdj_method	Method for adjusting p-values for multiple comparisons. See ?p.adjust for methods available. Default to FDR.
sortBy	Sort the output by p-value, gene-set name or sample names.

Details

Normalise the test perturbation scores generated by weight_ss_fc() through the permuted perturbation scores derived from the generate_permuted_scores() function. The mean absolute deviation(MAD) and median of perturbation scores for each pathway are firstly derived from the permuted perturbation scores. The test perturbation scores are then converted to robust z-scores using MADs and medians calculated.

Additionally, by assessing the proportion of permuted scores that are more extreme than the test perturbation score within each pathway, the permuted p-value of individual test perturbation scores will be computed.

Value

A data.frame

Examples

```
## Not run:
load(system.file("extdata", "gsTopology.rda", package = "sSNAPPY"))
data(metadata_example)
data(logCPM_example)
metadata_example <- dplyr::mutate(metadata_example, treatment = factor(
   treatment, levels = c("Vehicle", "E2+R5020", "R5020")))
ls <- weight_ss_fc(logCPM_example, metadata = metadata_example,
groupBy = "patient", sampleColumn = "sample", treatColumn = "treatment")

# compute raw gene-wise perturbation scores
genePertScore <- raw_gene_pert(ls$weighted_logFC, gsTopology)

# sum gene-wise perturbation scores to derive the pathway-level
# single-sample perturbation scores
pathwayPertScore <- pathway_pert(genePertScore, ls$weighted_logFC)

# simulate the null distribution of scores through sample permutation
permutedScore <- generate_permuted_scores(logCPM_example,
gsTopology = gsTopology, weight = ls$weight)

# normlise the test perturbation scores using the permutation results
normalisedScores <- normalise_by_permu(permutedScore, pathwayPertScore,
sortBy = "pvalue")

## End(Not run)
```

---

pathway_pert *Compute Single-sample Pathway-level Perturbation Score*

Description

Substract ssFC from the raw gene-level perturbation scores within each sample and sum gene-wise raw perturbation scores to derive single-sample perturbation scores for each pathway.

Usage

```
pathway_pert(genePertScore, weightedFC, drop = TRUE)
```

Arguments

genePertScore	List of gene-wise raw perturbation score matrices generated using function raw_gene_pert()
weightedFC	A matrix of weighted ssFC generated using function weight_ss_fc
drop	logic(1). Whether to drop pathways with all zero scores

Value

A data.frame with 3 columns: score (single-sample pathway-level perturbation score), sample, and gs_name (gene-set name)

References

Tarca AL, Draghici S, Khatri P, Hassan SS, Mittal P, Kim JS, Kim CJ, Kusanovic JP, Romero R. A novel signaling pathway impact analysis. Bioinformatics. 2009 Jan 1;25(1):75-82.

Examples

```
#compute weighted single sample logFCs
data(metadata_example)
data(logCPM_example)
metadata_example <- dplyr::mutate(metadata_example, treatment = factor(
   treatment, levels = c("Vehicle", "E2+R5020", "R5020")))
ls <- weight_ss_fc(logCPM_example, metadata = metadata_example,
groupBy = "patient", treatColumn = "treatment", sampleColumn = "sample")
# extract all the KEGG pathways
gsTopology <- retrieve_topology(database = "kegg", species = "hsapiens")
# compute raw gene-wise perturbation scores
genePertScore <- raw_gene_pert(ls$weighted_logFC, gsTopology)
# sum gene-wise perturbation scores to derive the pathway-level single-sample
# perturbation scores
pathwayPertScore <- pathway_pert(genePertScore, ls$weighted_logFC)
```

---

plot_community *Visualise the community structure in significantly perturbed gene-set network*

Description

Visualise the community structure in significantly perturbed gene-set network

Usage

```
plot_community(
  normalisedScores,
  gsTopology,
  gsAnnotation = NULL,
  colorBy = "community",
  communityMethod = c("louvain", "walktrap", "spinglass", "leading_eigen",
    "edge_betweenness", "fast_greedy", "label_prop", "leiden"),
  foldGSname = TRUE,
  foldafter = 2,
  labelFun = .rm_prefix,
  layout = c("fr", "dh", "gem", "graphopt", "kk", "lgl", "mds", "sugiyama"),
  markCommunity = "ellipse",
```

```
  markAlpha = 0.2,
  color_lg_title = NULL,
  edgeAlpha = 0.8,
  scale_edgeWidth = c(0.5, 3),
  edgeLegend = FALSE,
  scale_nodeSize = c(3, 6),
  nodeShape = 16,
  lb_size = 3,
  lb_color = "black",
  plotIsolated = FALSE,
  ...
)
```

Arguments

normalisedScores	A data.frame derived from normalise_by_permu
gsTopology	List of pathway topology matrices generated using retrieve_topology
gsAnnotation	A data.frame containing gene-sets categories for pathway annotation. Must contain the two columns: c("gs_name", "category"), where gs_name denotes gene-sets names that are matched to names of pathway topology matrices, and category records a higher level category for each pathway. If customized annotation is not provided, it will be assumed that the pathways were obtained from the KEGG database and inbuilt KEGG pathway annotation information will be used
colorBy	Can be any column with in the normalisedScores object, or the additional value "community".
communityMethod	A community detection method supported by igraph. See details for all methods available.
foldGSname	logical. Should long gene-set names be folded into two lines
foldafter	The number of words after which gene-set names should be folded. Defaults to 2
labelFun	function to manipulate or modify gene-set labels. By default, any database will be stripped from the prefix using a regex pattern
layout	The layout algorithm to apply. Accepted layouts are "fr", "dh", "gem", "graphopt", "kk", "lgl", '
markCommunity	character A geom_mark_* method supported by ggforce to annotate sets of nodes belonging to the same community. Either *NULL*, *ellipse*, *circle*, *hull*, *rect*
markAlpha	Transparency of annotation areas.
color_lg_title	Title for the color legend
edgeAlpha	Transparency of edges.
scale_edgeWidth	A numerical vector of length 2 to be provided to ggraph::scale_edge_width_continuous() for specifying the minimum and maximum edge widths after transformation.
edgeLegend	logical Should edge weight legend be shown

scale_nodeSize	A numerical vector of length 2 to be provided to ggplot2::scale_size() for specifying the minimum and maximum node sizes after transformation.
nodeShape	The shape to use for nodes
lb_size	Size of node text labels
lb_color	Color of node text labels
plotIsolated	logical(1) Should nodes not connected to any other nodes be plotted. Defaults to FALSE
...	Used to pass various potting parameters to ggforce::geom_mark_*()

Details

A community detection strategy specified by communityMethod will be applied to the pathway-pathway network, and communities will be annotated with the pathway category that had the highest number of occurrence, denoting the main biological processes perturbed in that community.

At the moment, only KEGG pathway categories are provided with the package, so if the provided normalisedScores contains perturbation scores of pathways derived from other databases, annotation of communities will not be performed unless pathway information is provided through the gsAnnotation object. The category information needs to be provided in a data.frame containing gs_name (gene-set names) and category (categorising the given pathways).

Plotting parameters accepted by geom_mark_* could be passed to the function to adjust the annotation area or the annotation label. See geom_mark_ellipse for more details.

Value

A ggplot2 object

Examples

```
load(system.file("extdata", "gsTopology.rda", package = "sSNAPPY"))
load(system.file("extdata", "normalisedScores.rda", package = "sSNAPPY"))
#Subset the first 10 rows of the normalisedScores data.frame as an example
subset <- normalisedScores[1:15,]
subset$status <- ifelse(subset$robustZ > 0, "Activated", "Inhibited")
# Color network plot nodes by the community they were assigned to and mark
# nodes belonging to the same community by ellipses
plot_community(subset, gsTopology, colorBy = "community",layout = "kk",
color_lg_title = "Community")

# Color network plot nodes by pathways' directions of changes and mark nodes
# belonging to the same community by ellipses
plot_community(subset, gsTopology, colorBy = "status",layout = "kk",
color_lg_title = "Direction of pathway perturbation")

# To change the colour and fill of `geom_mark_*` annotation, use any
# `scale_fill_*` and/or `scale_color_*`
# functions supported by `ggplot2`. For example:
p <- plot_community(subset, gsTopology, colorBy = "status",layout = "kk",
markCommunity = "rect",color_lg_title = "Direction of pathway perturbation")
p + ggplot2::scale_color_ordinal() + ggplot2::scale_fill_ordinal()
```

`plot_gene_contribution` *Plot genes' contribution to a pathway's perturbation as a heatmap*

Description

Plot individual genes' contributions to the pathway-level perturbation score

Usage

```
plot_gene_contribution(
  genePertMatr,
  mapEntrezID = NULL,
  topGene = 10,
  filterBy = c("mean", "sd", "max.abs"),
  tieMethod = "min",
  annotation_df = NULL,
  ...
)
```

Arguments

`genePertMatr`	A matrix of gene-wise perturbation scores corresponding to a pathway. An element of the output generated using function `raw_gene_pert()`
`mapEntrezID`	Optional. A `data.frame` matching genes' entrez IDs to another identifier with preferred labels. Must contain the columns: `"entrezid"` and `"mapTo"`
`topGene`	Numeric(1). The number of top genes to plot
`filterBy`	Filter top genes by the mean, variability (sd), maximum value, or maximum absolute values
`tieMethod`	Method for handling ties in ranking (i.e. in values returned by `filterBy`, two or many genes share the same value). Default to "min". See `?rank` for other options.
`annotation_df`	A `data.frame` for annotating heatmap columns. Must contain a "sample" column with sample names matching to the column names of the `genePertMatr`
`...`	Used to pass various potting parameters to `pheatmap::pheatmap()`

Details

The single-sample pathway-level perturbation score for a given pathway is derived from aggregating all the gene-wise perturbation scores of genes in that pathway. This function visualises individual pathway genes' perturbation scores as a heatmap to demonstrate pathway genes' contribution to a pathway perturbation.

Plotting of the heatmap is done through `pheatmap::pheatmap()` so all plotting parameters accepted by `pheatmap::pheatmap()` could also be passed to this function.

References

Kolde R (2019). *pheatmap: Pretty Heatmaps*. R package version 1.0.12, https://CRAN.R-project.org/package=pheatmap.

Examples

```
#compute weighted single sample logFCs
data(metadata_example)
data(logCPM_example)
metadata_example <- dplyr::mutate(metadata_example, treatment = factor(
   treatment, levels = c("Vehicle", "E2+R5020", "R5020")))
# compute single-sample logFCs for all treated samples
ls <- weight_ss_fc(logCPM_example, metadata = metadata_example,
groupBy = "patient", treatColumn = "treatment", sampleColumn = "sample")

# extract all the KEGG pathways
gsTopology <- retrieve_topology(database = "kegg", species = "hsapiens")

# compute raw gene-wise perturbation scores
genePertScore <- raw_gene_pert(ls$weighted_logFC, gsTopology)
# sum gene-wise perturbation scores to derive the pathway-level single-sample perturbation scores
pathwayPertScore <- pathway_pert(genePertScore, ls$weighted_logFC)

# Genes with top 10 mean absolute gene-wise perturbation scores in the
# Estrogen signaling pathway was visualised.
plot_gene_contribution(genePertScore$`kegg.Estrogen signaling pathway`,
filterBy = "mean", topGene = 10)

# Columns of the heatmap could be annotated by the pathway-level perturbation
# and treatments. Firstly, create a `data.frame` with the two annotation
# attributes and sample names matching the column names of the perturbation
# score matrix.
annotation_df <- dplyr::select(metadata_example, sample, treatment)
pathwayLevel <- dplyr::filter(pathwayPertScore,
gs_name == "kegg.Estrogen signaling pathway")
pathwayLevel$`pathway-level` <- ifelse(
pathwayLevel$score > 0, "Activated", "Inhibited")
annotation_df <- dplyr::left_join(
dplyr::select(pathwayLevel, sample, `pathway-level`),
annotation_df, unmatched = "drop")
# To make the gene labels more informative, also map genes' entrez id
# to chosen identifiers.
load(system.file("extdata", "entrez2name.rda", package = "sSNAPPY"))
plot_gene_contribution(genePertScore$`kegg.Estrogen signaling pathway`,
topGene = 10, filterBy = "mean", annotation_df = annotation_df,
mapEntrezID = entrez2name)

# Plotting parameters accepted by `pheatmap::pheatmap()` could be passed to
# this function to customise the plot. For example, changin the color of annotations
plot_gene_contribution(genePertScore$`kegg.Estrogen signaling pathway`,
topGene = 10, filterBy = "mean", annotation_df = annotation_df,
mapEntrezID = entrez2name, annotation_colors = list(
```

```
treatment = c(R5020 = "black", `E2+R5020` = "white"),
`pathway-level` = c(Activated = "darkred", Inhibited = "lightskyblue")))
```

---

plot_gs2gene *Plot pathways and genes contained in them as a network*

Description

Plot pathways and genes contained in them as a network

Usage

```
plot_gs2gene(
  normalisedScores,
  gsTopology,
  geneFC = NULL,
  mapEntrezID = NULL,
  colorGsBy = NULL,
  foldGSname = TRUE,
  foldafter = 2,
  labelFun = .rm_prefix,
  filterGeneBy = 2,
  layout = c("fr", "dh", "gem", "graphopt", "kk", "lgl", "mds", "sugiyama"),
  edgeColor = "darkgrey",
  edgeAlpha = 0.8,
  edgeArc = 0.5,
  geneNodeSize = 3,
  geneNodeShape = 17,
  geneNameFace = c("italic", "plain", "bold", "bold-italic"),
  geneNameColor = "grey30",
  geneNameSize = 3,
  labelGene = TRUE,
  gsNodeSize = 2,
  gsNodeShape = 21,
  gsNodeStroke = 0.5,
  gsNodeOutline = "white",
  gsNameSize = 6,
  gsNameColor = "black",
  geneLegTitle = "Mean logFC",
  gsLegTitle = colorGsBy,
  maxOverlaps = 10,
  ...
)
```

Arguments

normalisedScores	A data.frame derived from the normalise_by_permu() function. Only gene-sets of interest should be included
gsTopology	List of pathway topology matrices generated using function retrieve_topology()
geneFC	An optional named vector of pathways' fold changes
mapEntrezID	Optional. A data.frame matching genes' entrez IDs to another identifier with preferred labels. Must contain the columns: "entrezid" and "mapTo"
colorGsBy	Column within normalisedScores to color gene-set/pathway nodes by
foldGSname	logical. Should long gene-set names be folded into two lines
foldafter	The number of words after which gene-set names should be folded.
labelFun	function to manipulate or modify gene-set labels. By default, any database will be stripped from the prefix using a regex pattern
filterGeneBy	Filtration cut-off applied to genes' connectivity (ie. how many pathways was a gene involved in).
layout	The layout algorithm to apply. Accepts all layout supported by igraph.
edgeColor, edgeAlpha	Color and transparency of edges
edgeArc	The bend of edges. 1 approximates a semi-circle whilst 0 will give a straight line.
geneNodeSize, geneNodeShape	Size and shape for gene nodes
geneNameSize, geneNameColor, geneNameFace	Size, color and fontface to use for gene labels
labelGene	logical(1) Should the gene names be included
gsNodeSize	Size for gene-set/pathway nodes
gsNodeShape	Shape for gene-set/pathway nodes. Should be a shape with a fill parameter, such as 21:25
gsNodeStroke, gsNodeOutline	Border thickness and color for gene-set/pathway nodes
gsNameSize, gsNameColor	Size and color of gene-set/pathway labels
geneLegTitle	character(1). Legend title for gene nodes
gsLegTitle	character(1) Legend title for gene-set/pathway nodes
maxOverlaps	passed to geom_node_text
...	Not used

Details

Taking the perturbation scores of a list of gene-sets derived from normalise_by_permu() as input, this function matches gene-sets to their associated genes by utilizing information from pathway topology matrices.

If providing logFC values as a named vector, the names must be entrezgene IDs in the format of "ENTREZID:XXXX" for compatibility with the values returned by `retrieve_topology()`. If not providing this vector, only genes associated with two or more pathways will be added to the plot, however, it should be noted that if omitting this vector, network plots can easily become unmanageable.

Users can also choose to provide a `mapEntrezID` `data.frame` to match entrezgene IDs to their chosen identifiers. The `data.frame` should contain the columns: `"entrezid"` and `"mapTo"`.

If `geneFC` is provided, gene nodes will be colored by values provided, otherwise all gene nodes will drawn in grey.

Since some gene-sets could can contain hundreds of genes, it is not recommended to plot all genes. If `mapEntrezID` `data.frame` is provided, only genes included in that `data.frame` will be used in the plot.

It is strongly recommended to filter genes using some criteria, such as those with the largest magnitude of change. If all pathway genes are desired, please consider setting `labelGene` to FALSE to remove gene names.

Value

A ggplot2 object

Examples

```
load(system.file("extdata", "gsTopology.rda", package = "sSNAPPY"))
load(system.file("extdata", "normalisedScores.rda", package = "sSNAPPY"))

# Subset pathways significantly perturbed in sample R5020_N2_48
subset <- dplyr::filter(normalisedScores, adjPvalue < 0.05, sample == "R5020_N2_48")
subset$response <- ifelse(subset$robustZ > 0, "Activated", "Inhibited")

# Color gene-sets nodes by robust z-scores.
plot_gs2gene(
  subset, gsTopology, colorGsBy = "robustZ", labelGene = FALSE, geneNodeSize = 1,
  gsNodeSize = 4
) + scale_fill_gradient2()
# When fold-changes are not provided, gene nodes are colored grey.

# To color genes by their direction of change, firstly compute single-sample logFC
data(logCPM_example)
data(metadata_example)
metadata_example <- dplyr::mutate(metadata_example, treatment = factor(
   treatment, levels = c("Vehicle", "E2+R5020", "R5020")))
ls <- weight_ss_fc(
  logCPM_example, metadata = metadata_example,
  groupBy = "patient", treatColumn = "treatment",
  sampleColumn = "sample"
)
# Provide fold-changes of sample R5020_N2_48 as a named vector
plot_gs2gene(
  subset, gsTopology, geneFC = ls$weighted_logFC[,"R5020_N2_48"],
  colorGsBy = "response", labelGene = FALSE
```

```
) + scale_colour_gradient2()

# By default, the function only include genes involved in at least 2 pathways,
# which can be overwritten by the `filterGeneBy` parameter. But there are still
# a large number of genes, making the plot cumbersome. Instead, only include
# fold-changes of genes within the top 500 absolute values for fold-change
top500 <- rank(1/abs(ls$weighted_logFC[,"R5020_N2_48"])) <= 500
fcByDir <- ifelse(ls$weighted_logFC[top500,"R5020_N2_48"] > 0, "Up-Regulated", "Down-Regulated")
plot_gs2gene(subset, gsTopology, geneFC = fcByDir, colorGsBy = "response") +
  scale_fill_manual(values = c("darkred", "lightskyblue")) +
  scale_colour_manual(values = c("red", "blue"))

# To make the gene labels more informative, map genes' entrez id to chosen identifiers.
load(system.file("extdata", "entrez2name.rda", package = "sSNAPPY"))
plot_gs2gene(
  subset, gsTopology, geneFC = fcByDir, mapEntrezID = entrez2name,
  colorGsBy = "response", gsNodeSize = 4
) +
  scale_fill_manual(values = c("darkred", "lightskyblue"), name = "Pathway") +
  scale_colour_manual(values = c("blue", "red"), name = "Gene\nDirection")
```

`plot_gs_network` *Plot significantly perturbed gene-sets as a network*

Description

Plot significantly perturbed gene-sets as a network

Usage

```
plot_gs_network(
  normalisedScores,
  gsTopology,
  colorBy = NULL,
  foldGSname = TRUE,
  foldafter = 2,
  labelFun = .rm_prefix,
  layout = c("fr", "dh", "gem", "graphopt", "kk", "lgl", "mds", "sugiyama"),
  edgeAlpha = 0.8,
  edgeWidthScale = c(0.5, 3),
  edgeLegend = FALSE,
  nodeSizeScale = c(3, 6),
  nodeShape = 16,
  showLegend = TRUE,
  gsLegTitle = NULL,
  gsNameSize = 3,
  gsNameColor = "black",
  plotIsolated = FALSE,
```

```
  maxOverlaps = 10,
  ...
)
```

Arguments

normalisedScores	A data.frame of pathway perturbation scores derived from the normalise_by_permu() function
gsTopology	List of pathway topology matrices generated using function retrieve_topology()
colorBy	Choose to color nodes either by *robustZ* or *pvalue*. A column must exist in the normalisedScores data.frame for the chosen parameter
foldGSname	logical(1) Should long gene-set names fold across multiple lines
foldafter	The number of words after which gene-set names should be folded. Defaults to 2
labelFun	function to manipulate or modify gene-set labels. By default, any database will be stripped from the prefix using a regex pattern
layout	The layout algorithm to apply. Accept all layout supported by igraph
edgeAlpha	numeric(1) Transparency of edges. Default to 0.8
edgeWidthScale	A numerical vector of length 2 to be provided to ggraph::scale_edge_width_continuous() for specifying the minimum and maximum edge widths after transformation. Defaults to c(0.5, 3)
edgeLegend	logical(1) Should edge weight legend be shown
nodeSizeScale	A numeric vector of length 2 to be provided to ggplot2::scale_size() for specifying the minimum and maximum node sizes after transformation. Defaulted to c(3,6)
nodeShape	Shape of nodes
showLegend	logical(1) Should color legend be shown
gsLegTitle	Optional title for the color legend
gsNameSize	Size of node text labels
gsNameColor	Color of node text labels
plotIsolated	logical(1) Should nodes not connected to any other nodes be plotted. Defaults to FALSE
maxOverlaps	passed to geom_node_text
...	Not used

Value

A ggplot2 object

Examples

```
load(system.file("extdata", "gsTopology.rda", package = "sSNAPPY"))
load(system.file("extdata", "normalisedScores.rda", package = "sSNAPPY"))

# Subset pathways significantly perturbed in sample R5020_N2_48
subset <- dplyr::filter(
  normalisedScores, adjPvalue < 0.05, sample == "R5020_N2_48"
)
subset[["status"]] <- ifelse(subset[["robustZ"]]> 0, "Activated", "Inhibited")


# Color network plot nodes by status
plot_gs_network(subset, gsTopology,
colorBy = "status", layout = "dh",
gsLegTitle = "Direction of pathway Perturbation")

# Color network plot nodes by p-values
plot_gs_network(subset, gsTopology, layout = "dh",
colorBy = "pvalue", gsLegTitle = "P-value")
```

---

raw_gene_pert *Compute Gene-wise Perturbation Score*

Description

Propagate weighted single sample logFCs down the pathway topologies to compute gene-wise perturbation score per gene per sample per pathway

Usage

```
raw_gene_pert(weightedFC, gsTopology)
```

Arguments

weightedFC	A matrix of weighted single sample logFCs derived from function `weight_ss_fc()`
gsTopology	List of pathway topology matrices generated using function `retrieve_topology()`

Details

This function use the algorithm adopted from SPIA (see citation) to integrate genes' changes in expression and gene-gene interaction to compute gene-wise perturbation score per gene per sample per pathway. The rownames of the weighted single sample logFC matrix and the pathway topology matrices must use the same type of gene identifier (ie. entrez ID).

Pathways with zero perturbation scores across all genes and samples will be dropped from the output.

Value

A list where each element is a matrix corresponding to a pathway. Each column of an element corresponds to a sample, and each row corresponds to a pathway gene.

References

Tarca AL, Draghici S, Khatri P, Hassan SS, Mittal P, Kim JS, Kim CJ, Kusanovic JP, Romero R. A novel signaling pathway impact analysis. Bioinformatics. 2009 Jan 1;25(1):75-82.

Examples

```
#compute weighted single sample logFCs
data(metadata_example)
data(logCPM_example)
metadata_example <- dplyr::mutate(metadata_example, treatment = factor(
   treatment, levels = c("Vehicle", "E2+R5020", "R5020")))
ls <- weight_ss_fc(logCPM_example, metadata = metadata_example,
groupBy = "patient", treatColumn = "treatment", sampleColumn = "sample")
# extract all the KEGG pathways
gsTopology <- retrieve_topology(database = "kegg", species = "hsapiens")
# compute raw gene-wise perturbation scores
genePertScore <- raw_gene_pert(ls$weighted_logFC, gsTopology)
```

---

retrieve_topology *Retrieve pathway topology as weighted adjacency matrices*

Description

Retrieve pathway topology matrices and convert them to normalized weighted directed adjacency matrices describing gene signaling networks.

Usage

```
retrieve_topology(
  database = c("kegg", "wikipathways", "reactome"),
  species = c("hsapiens", "athaliana", "btaurus", "celegans", "cfamiliaris",
    "dmelanogaster", "drerio", "ecoli", "ggallus", "mmusculus", "rnorvegicus",
    "scerevisiae", "sscrofa", "xlaevis"),
  keyword = NULL,
  beta = NULL
)
```

Arguments

database	A character vector of supported databases.
species	One of the supported species. Currently support `c("hsapiens", "athaliana", "btaurus", "celegans", "cfamiliaris", "dmelanogaster", "drerio", "ecoli", "ggallus", "mmusculus","rnorvegicus", "scerevisiae", "sscrofa", "xlaevis")`

keyword	Optional. Areas of interests as a character vector.
beta	Optional. A named numeric vector of weights to be assigned to each type of gene/protein relation type.See details for more information.

Details

This function takes pathway topology information retrieved using `graphite` and converts these to normalized weighted directed adjacency matrices describing the gene signaling network, which can be used to compute gene-wise and pathway-level perturbation score through the scoring algorithm derived from the *SPIA* algorithm. See cited document for more details.

The `database` parameter may specify multiple databases but only one species can be provided to the `species` parameter.

Users can provide areas of interests as keywords, which will be matched to pathway names from chosen databases to subset pathways. Cases will be ignored in string matching.

The beta parameter specifies weights to be assigned to each type of gene-gene interaction. It should be a named numeric vector of length 25, whose names must be: `c("activation","compound","binding/association", "expression","inhibition","activation_phosphorylation","phosphorylation", "indirect","inhibition_pho "dissociation","dephosphorylation","activation_dephosphorylation", "state","activation_indirect","in "expression_indirect","indirect_inhibition","repression", "binding/association_phosphorylation","dis "indirect_phosphorylation")`.

If unspecified, beta will be set as an integer vector with: a) values of 1 for interactions which match 'expression' or 'activation'; b) values of -1 for interactions which match 'repression' or 'inhibition'; and c) 0 elsewhere.

The retrieved topology matrices will be processed as decribed by *SPIA* to: 1) scale gene-gene interactions by the number of downstream genes and 2) subtract an identity matrix of the same size from the topology matrix.

The converted weighted adjacent matrices will be stored in a list. We recommend users to store the returned list as a file so this step only needs to be performed once for each database.

Value

A list where each element is a weighted directed adjacency matrix corresponding to a pathway

References

Tarca AL, Draghici S, Khatri P, Hassan SS, Mittal P, Kim JS, Kim CJ, Kusanovic JP, Romero R. A novel signaling pathway impact analysis. Bioinformatics. 2009 Jan 1;25(1):75-82. Sales, G., Calura, E., Cavalieri, D. et al. graphite - a Bioconductor package to convert pathway topology to gene network. BMC Bioinformatics 13, 20 (2012).

Examples

```
# retrieve pathway topology matrices of all KEGG pathway
gsTopology <- retrieve_topology(database = "kegg", species = "hsapiens")

# If only interested in selected pathways, specify the areas of interest as
# keywords
gsTopology <- retrieve_topology(database = "kegg",
```

```
keyword = c("metabolism", "signaling"), species = "hsapiens")
```

---

sSNAPPY *sSNAPPY: A package for testing directional single sample pathway perturbation*

Description

A package for testing directional single sample pathway perturbation

---

weight_ss_fc *Compute weighted single sample LogFC from normalised logCPM*

Description

Compute weighted single sample logFC for each treated samples using normalized logCPM values. Fit a lowess curve on variances ~ mean of logCPM values, and use it to predict gene-wise weights. The weighted single sample logFC are ready to be used for computing perturbation scores.

Usage

```
weight_ss_fc(
  expreMatrix,
  metadata = NULL,
  sampleColumn,
  treatColumn,
  groupBy,
  prefix = "ENTREZID:"
)

## S4 method for signature 'matrix'
weight_ss_fc(
  expreMatrix,
  metadata = NULL,
  sampleColumn,
  treatColumn,
  groupBy,
  prefix = "ENTREZID:"
)

## S4 method for signature 'data.frame'
weight_ss_fc(
  expreMatrix,
  metadata = NULL,
```

```
  sampleColumn,
  treatColumn,
  groupBy,
  prefix = "ENTREZID:"
)

## S4 method for signature 'DGEList'
weight_ss_fc(
  expreMatrix,
  metadata = NULL,
  sampleColumn,
  treatColumn,
  groupBy,
  prefix = "ENTREZID:"
)

## S4 method for signature 'SummarizedExperiment'
weight_ss_fc(
  expreMatrix,
  metadata = NULL,
  sampleColumn,
  treatColumn,
  groupBy,
  prefix = "ENTREZID:"
)
```

Arguments

`expreMatrix`	`matrix` or `data.frame` of logCPM, or `DGEList`/ `SummarizedExperiment` storing gene expression counts and sample metadata. Feature names need to be in entrez IDs, and column names need to be sample names
`metadata`	Sample metadata `data.frame` as described in the details section.
`sampleColumn`	Name of the column in the `metadata` containing column names of the `expreMatrix`
`treatColumn`	Name of the column in the `metadata` containing treatment information. The column must be a factor with the reference level set to be the control treatment.
`groupBy`	Name of the column in the `metadata` containing information for how samples are matched in pairs (eg. patient).
`prefix`	Character(1). Prefix to be add to the `rownames` of the `expreMatrix`. Default to `ENTREZID:` to align with the format that topology matrices will be retrieved in, but can be adjusted according to users' needs.

Details

This function computes weighted single-sample logFC (ssFC) from normalised logCPM values, used for computing single-sample perturbation scores.

Genes with low expression tend to have larger variances, which will introduce biases to the ssFC computed. Therefore, a lowess curve will be fitted to estimate the relationship between the variances

and the mean of logCPM, and the relationship will be used to estimate the variance of each gene-wise mean logCPM value. Gene-wise weights, which are defined to be the inverse of predicted variances that are scaled to have a sum of 1, will then be multiplied to ssFC to down-weight genes with low counts.

It is assumed that the genes with extremely low counts have been removed and the count matrix has been normalised prior to the logCPM matrix was derived. Row names of the matrix must be in genes' entrez IDs to align with the fomrat of pathway topology matrices.

If a S4 object of `DGEList` or `SummarizedExperiment` is provided as input to `expreMatrix`, the gene expression matrix will be extracted from it and converted to a logCPM matrix. Sample metadata will also be extracted from the same S4 object unless otherwise specified.

Provided sample metadata should have the same number of rows as the number of columns in the logCPM matrix and must contain the a column for treatment, one for sample names and a column for how samples should be matched into pairs.

The treatment column in the sample metadata must be a factor with the reference level set to be the control condition.

Value

A list with two elements: $weight gene-wise weights; $weighted_logFC weighted single sample logFC matrix

Examples

```
# Inspect metadata data frame to make sure it has treatment, sample
# and patient columns
data(metadata_example)
data(logCPM_example)
# Set the treatment column to be a factor where the reference is the control
#treatment
metadata_example <- dplyr::mutate(metadata_example, treatment = factor(
   treatment, levels = c("Vehicle", "E2+R5020", "R5020")))
ls <- weight_ss_fc(logCPM_example, metadata = metadata_example,
 sampleColumn = "sample", groupBy = "patient", treatColumn = "treatment")
```

Index