Package 'BioNERO'

October 15, 2023

Type Package

Title Biological Network Reconstruction Omnibus

Version 1.8.7

Description BioNERO aims to integrate all aspects of biological network inference in a single package, including data preprocessing, exploratory analyses, network inference, and analyses for biological interpretations. BioNERO can be used to infer gene coexpression networks (GCNs) and gene regulatory networks (GRNs) from gene expression data. Additionally, it can be used to explore topological properties of protein-protein interaction (PPI) networks. GCN inference relies on the popular WGCNA algorithm. GRN inference is based on the ``wisdom of the crowds'' principle, which consists in inferring GRNs with multiple algorithms (here, CLR, GENIE3 and ARACNE) and calculating the average rank for each interaction pair. As all steps of network analyses are included in this package, BioNERO makes users avoid having to learn the syntaxes of several packages and how to communicate between them. Finally, users can also identify consensus modules across independent expression sets and calculate intra and interspecies module preservation statistics between different networks.

Depends R (>= 4.1)

License GPL-3

Encoding UTF-8

LazyData false

URL https://github.com/almeidasilvaf/BioNERO

BugReports https://github.com/almeidasilvaf/BioNERO/issues

biocViews Software, GeneExpression, GeneRegulation, SystemsBiology, GraphAndNetwork, Preprocessing, Network, NetworkInference

Imports WGCNA, dynamicTreeCut, ggdendro, matrixStats, sva, RColorBrewer, ComplexHeatmap, ggplot2, rlang, ggrepel, patchwork, reshape2, igraph, ggnetwork, intergraph, NetRep, stats, grDevices, utils, methods, BiocParallel, minet, GENIE3, SummarizedExperiment

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0), BiocStyle, DESeq2, networkD3, covr

VignetteBuilder knitr

Config/testthat/edition 3

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

git_branch RELEASE_3_17

git_last_commit cbfb2bf

git_last_commit_date 2023-08-21

Date/Publication 2023-10-15

Author Fabricio Almeida-Silva [cre, aut] (<`https://orcid.org/0000-0002-5314-2964`>), Thiago Venancio [aut] (<`https://orcid.org/0000-0002-2215-8082`>)

Maintainer Fabricio Almeida-Silva <`fabricio_almeidasilva@hotmail.com`>

R topics documented:

`check_SFT` *Check scale-free topology fit for a given network*

Description

Check scale-free topology fit for a given network

Usage

```
check_SFT(edgelist, net_type = "gcn")
```

Arguments

`edgelist`	Edge list as a data frame containing node 1, node 2 and edge weight.
`net_type`	Type of biological network. One of "gcn", "grn", or "ppi". Default: gcn.

Value

A list with SFT fit statistics and a message indicating if the network is scale-free.

Examples

```
set.seed(1)
exp <- t(matrix(rnorm(10000), ncol=1000, nrow=200))
rownames(exp) <- paste0("Gene", 1:nrow(exp))
colnames(exp) <- paste0("Sample", 1:ncol(exp))
cormat <- cor(t(exp))
edges <- cormat_to_edgelist(cormat)
edges <- edges[abs(edges$Weight) > 0.10, ]
check_SFT(edges)
```

`consensus_modules` *Identify consensus modules across independent data sets*

Description

Identify consensus modules across independent data sets

Usage

```
consensus_modules(
  exp_list,
  metadata,
  power,
  cor_method = "spearman",
  net_type = "signed hybrid",
  module_merging_threshold = 0.8,
  TOM_type = NULL,
  verbose = FALSE
)
```

Arguments

`exp_list`	A list containing the expression data frames with genes in row names and samples in column names or ‘SummarizedExperiment‘ objects. The list can be created by using `list(exp1, exp2, ..., expn)`.
`metadata`	A data frame containing sample names in row names and sample annotation in the first column. Ignored if ‘exp_list‘ is a list of ‘SummarizedExperiment‘ objects, since the function will extract colData.
`power`	Numeric vector of beta power for each expression set as calculated by `consensus_SFT_fit`.
`cor_method`	Correlation method used for network reconstruction. One of "spearman" (default), "biweight", or "pearson".
`net_type`	Network type. One of "signed hybrid" (default), "signed" or "unsigned".

`module_merging_threshold`	Correlation threshold to merge similar modules into a single one. Default: 0.8.
`TOM_type`	Character indicating the type of Topological Overlap Matrix to (TOM) create. One of 'unsigned', 'signed', 'signed Nowick', 'unsigned 2', 'signed 2', and 'signed Nowick 2'. By default, TOM type is automatically selected based on network type.
`verbose`	Logical indicating whether to display progress messages or not. Default: FALSE.

Value

A list containing 4 elements:

consMEs Consensus module eigengenes

exprSize Description of the multi-set object returned by the function `WGCNA::checkSets`

sampleInfo Metadata for each expression set

genes_cmodules Data frame of genes and consensus modules

dendro_plot_objects Objects to be used in dendrogram plotting

Examples

```
set.seed(12)
data(zma.se)
filt.zma <- filter_by_variance(zma.se, n=500)
zma.set1 <- filt.zma[, sample(colnames(filt.zma), size=20, replace=FALSE)]
zma.set2 <- filt.zma[, sample(colnames(filt.zma), size=20, replace=FALSE)]
list.sets <- list(zma.set1, zma.set2)
# SFT power previously identified with consensus_SFT_fit()
cons_mod <- consensus_modules(list.sets, power = c(11, 13),
                              cor_method = "pearson")
```

`consensus_SFT_fit` *Pick power to fit networks to scale-free topology*

Description

Pick power to fit networks to scale-free topology

Usage

```
consensus_SFT_fit(
  exp_list,
  setLabels = NULL,
  metadata = NULL,
  cor_method = "spearman",
  net_type = "signed hybrid",
  rsquared = 0.8
)
```

Arguments

`exp_list`	A list of expression data frames or SummarizedExperiment objects. If input is a list of data frames, row names must correspond to gene IDs and column names to samples. The list can be created with `list(exp1, exp2, ..., expn)`.
`setLabels`	Character vector containing labels for each expression set.
`metadata`	A data frame containing sample names in row names and sample annotation in the first column. Ignored if ‘exp_list‘ is a list of ‘SummarizedExperiment‘ objects, since the function will extract colData.
`cor_method`	Correlation method used for network reconstruction. One of "spearman" (default), "biweight", or "pearson".
`net_type`	Network type. One of "signed hybrid" (default), "signed" or "unsigned".
`rsquared`	Minimum R squared to consider the network similar to a scale-free topology. Default is 0.8.

Value

A list of 2 elements:

power Numeric vector of optimal beta powers to fit networks to SFT

plot A ggplot object displaying main statistics of the SFT fit test

Examples

```
set.seed(12)
data(zma.se)
filt.zma <- filter_by_variance(zma.se, n=500)
zma.set1 <- filt.zma[, sample(colnames(filt.zma), size=20, replace=FALSE)]
zma.set2 <- filt.zma[, sample(colnames(filt.zma), size=20, replace=FALSE)]
list.sets <- list(zma.set1, zma.set2)
cons_sft <- consensus_SFT_fit(list.sets, setLabels = c("Maize1", "Maize2"),
                              cor_method = "pearson")
```

`consensus_trait_cor` *Correlate set-specific modules and consensus modules to sample information*

Description

Correlate set-specific modules and consensus modules to sample information

Usage

```
consensus_trait_cor(consensus, cor_method = "pearson", metadata_cols = NULL)
```

Arguments

`consensus`	Consensus network returned by `consensus_modules`.
`cor_method`	Correlation method to be used. One of 'spearman' or 'pearson'. Default: 'pearson'.
`metadata_cols`	A vector (either numeric or character) indicating which columns should be extracted from column metadata if **exp** is a 'SummarizedExperiment' object. The vector can contain column indices (numeric) or column names (character). By default, all columns are used.

Value

Data frame of consensus module-trait correlations and p-values, with the following variables:

trait Factor, trait name. Each trait corresponds to a variable of the sample metadata (if numeric) or levels of a variable (if categorical).

ME Factor, module eigengene.

cor Numeric, correlation.

pvalue Numeric, correlation P-values.

group Character, name of the metadata variable.

Examples

```
set.seed(12)
data(zma.se)
filt.zma <- filter_by_variance(zma.se, n=500)
zma.set1 <- filt.zma[, sample(colnames(filt.zma), size=20, replace=FALSE)]
zma.set2 <- filt.zma[, sample(colnames(filt.zma), size=20, replace=FALSE)]
list.sets <- list(zma.set1, zma.set2)
# SFT power previously identified with consensus_SFT_fit()
consensus <- consensus_modules(list.sets, power = c(11, 13),
                               cor_method = "pearson")
consensus_trait <- consensus_trait_cor(consensus, cor_method = "pearson")
```

`cormat_to_edgelist` *Transform a correlation matrix to an edge list*

Description

Transform a correlation matrix to an edge list

Usage

```
cormat_to_edgelist(matrix)
```

Arguments

`matrix`	Symmetrical correlation matrix.

Value

A 2-column data frame containing node 1, node 2 and edge weight.

Examples

```
data(filt.se)
cor_mat <- cor(t(SummarizedExperiment::assay(filt.se)))
edgelist <- cormat_to_edgelist(cor_mat)
```

`detect_communities` *Detect communities in a network*

Description

Detect communities in a network

Usage

```
detect_communities(edgelist, method = igraph::cluster_infomap, directed = TRUE)
```

Arguments

`edgelist`	Data frame containing the network as an edge list. First column must be node 1 and second column must be node 2. Additional columns will be interpreted as edge attributes and will be modified by this function.
`method`	igraph function to be used for community detection. Available functions are cluster_infomap, cluster_edge_betweenness, cluster_fast_greedy, cluster_walktrap, cluster_spinglass, cluster_leading_eigen, cluster_louvain, and cluster_label_prop. Default is cluster_infomap.
`directed`	Logical indicating whether the network is directed (GRN only) or not (GCN and PPI networks). Default: TRUE.

Value

A data frame containing node names in the first column, and communities to which nodes belong in the second column.

Author(s)

Fabricio Almeida-Silva

See Also

`cluster_infomap`, `cluster_edge_betweenness`, `cluster_fast_greedy`, `cluster_walktrap`, `cluster_spinglass`, `cluster_leading_eigen`, `cluster_louvain`, `cluster_label_prop`

Examples

```
data(filt.se)
tfs <- sample(rownames(filt.se), size=50, replace=FALSE)
grn_edges <- grn_infer(filt.se, method = "clr", regulators = tfs)
com <- detect_communities(grn_edges, directed=TRUE)
```

`dfs2one` *Combine multiple expression tables (.tsv) into a single data frame*

Description

This function reads multiple expression tables (.tsv files) in a directory and combines them into a single gene expression data frame.

Usage

```
dfs2one(mypath, pattern = ".tsv$")
```

Arguments

mypath	Path to directory containing .tsv files. Files must have the first column in common, e.g. "Gene_ID". Rows are gene IDs and columns are sample names.
pattern	Pattern contained in each expression file. Default is '.tsv$', which means that all files ending in '.tsv' in the specified directory will be considered expression files.

Value

Data frame with gene IDs as row names and their expression values in each sample (columns).

Author(s)

Fabricio Almeida-Silva

Examples

```
# Simulate two expression data frames of 100 genes and 30 samples
genes <- paste0(rep("Gene", 100), 1:100)
samples1 <- paste0(rep("Sample", 30), 1:30)
samples2 <- paste0(rep("Sample", 30), 31:60)
exp1 <- cbind(genes, as.data.frame(matrix(rnorm(100*30),nrow=100,ncol=30)))
exp2 <- cbind(genes, as.data.frame(matrix(rnorm(100*30),nrow=100,ncol=30)))
colnames(exp1) <- c("Gene", samples1)
colnames(exp2) <- c("Gene", samples2)

# Write data frames to temporary files
tmpdir <- tempdir()
tmp1 <- tempfile(tmpdir = tmpdir, fileext = ".exp.tsv")
```

```
tmp2 <- tempfile(tmpdir = tmpdir, fileext = ".exp.tsv")
write.table(exp1, file=tmp1, quote=FALSE, sep="\t")
write.table(exp2, file=tmp2, quote=FALSE, sep="\t")

# Load the files into one
exp <- dfs2one(mypath = tmpdir, pattern=".exp.tsv")
```

enrichment_analysis *Perform overrepresentation analysis for a set of genes*

Description

Perform overrepresentation analysis for a set of genes

Usage

```
enrichment_analysis(
  genes,
  background_genes,
  annotation,
  column = NULL,
  correction = "BH",
  p = 0.05,
  min_setsize = 10,
  max_setsize = 500,
  bp_param = BiocParallel::SerialParam()
)
```

Arguments

genes	Character vector containing genes for overrepresentation analysis.
background_genes	Character vector of genes to be used as background for the overrepresentation analysis.
annotation	Annotation data frame with genes in the first column and functional annotation in the other columns. This data frame can be exported from Biomart or similar databases.
column	Column or columns of **annotation** to be used for enrichment. Both character or numeric values with column indices can be used. If users want to supply more than one column, input a character or numeric vector. Default: all columns from **annotation**.
correction	Multiple testing correction method. One of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr" or "none". Default is "BH".
p	P-value threshold. P-values below this threshold will be considered significant. Default: 0.05.

`min_setsize`	Numeric indicating the minimum gene set size to be considered. Gene sets correspond to levels of each variable in **annotation**). Default: 10.
`max_setsize`	Numeric indicating the maximum gene set size to be considered. Gene sets correspond to levels of each variable in **annotation**). Default: 500.
`bp_param`	BiocParallel back-end to be used. Default: BiocParallel::SerialParam()

Value

A data frame of overrepresentation results with the following variables:

term character, functional term ID/name.

genes numeric, intersection length between input genes and genes in a particular functional term.

all numeric, number of all genes in a particular functional term.

pval numeric, P-value for the hypergeometric test.

padj numeric, P-value adjusted for multiple comparisons using the method specified in parameter **adj**.

category character, name of the grouping variable (i.e., column name of **annotation**).

Author(s)

Fabricio Almeida-Silva

Examples

```
data(filt.se)
data(zma.interpro)
genes <- rownames(filt.se)[1:50]
background_genes <- rownames(filt.se)
annotation <- zma.interpro
# Using p = 1 to show all results
enrich <- enrichment_analysis(genes, background_genes, annotation, p = 1)
```

`exp2gcn` *Reconstruct gene coexpression network from gene expression*

Description

Reconstruct gene coexpression network from gene expression

Usage

```
exp2gcn(
  exp,
  net_type = "signed",
  module_merging_threshold = 0.8,
  SFTpower = NULL,
  cor_method = "spearman",
  verbose = FALSE
)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
`net_type`	Network type. One of ’signed’, ’signed hybrid’ or ’unsigned’. Default: ’signed’.
`module_merging_threshold`	Correlation threshold to merge similar modules into a single one. Default: 0.8.
`SFTpower`	SFT power generated by the function `SFT_fit`.
`cor_method`	Correlation method. One of "pearson", "biweight" or "spearman". Default is "spearman".
`verbose`	Logical indicating whether to display progress messages or not. Default: FALSE.

Value

List containing:

- Adjacency matrix
- Data frame of module eigengenes
- Data frame of genes and their corresponding modules
- Data frame of intramodular connectivity
- Correlation matrix
- Parameters used for network reconstruction
- Objects to plot the dendrogram in `plot_dendro_and_colors`.

Author(s)

Fabricio Almeida-Silva

See Also

`adjacency.fromSimilarity`,`TOMsimilarity`,`standardColors`,`labels2colors`,`moduleEigengenes`,`plotEigengeneNetw`
`cutreeDynamicTree`

Examples

```
data(filt.se)
# The SFT fit was previously calculated and the optimal power was 16
gcn <- exp2gcn(filt.se, SFTpower = 18, cor_method = "pearson")
```

exp2grn *Infer gene regulatory network from expression data*

Description

Infer gene regulatory network from expression data

Usage

```
exp2grn(
  exp,
  regulators = NULL,
  eps = 0,
  estimator_aracne = "spearman",
  estimator_clr = "pearson",
  remove_zero = TRUE,
  nsplit = 10,
  ...
)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a 'SummarizedExperiment' object.
`regulators`	A character vector of regulators (e.g., transcription factors or miRNAs). All regulators must be included in 'exp'.
`eps`	Numeric value indicating the threshold used when removing an edge: for each triplet of nodes (i,j,k), the weakest edge, say (ij), is removed if its weight is below min(ik),(jk) - eps. Default: 0.
`estimator_aracne`	Entropy estimator to be used in ARACNE inference. One of "mi.empirical", "mi.mm", "mi.shrink", "mi.sg", "pearson", "spearman", or "kendall". Default: "spearman".
`estimator_clr`	Entropy estimator to be used in CLR inference. One of "mi.empirical", "mi.mm", "mi.shrink", "mi.sg", "pearson", "spearman", or "kendall". Default: "pearson".
`remove_zero`	Logical indicating whether to remove edges whose weight is exactly zero. Zero values indicate edges that were removed by ARACNE. Default: TRUE.
`nsplit`	Number of groups in which the edge list will be split. Default: 10.
`...`	Additional arguments passed to 'GENIE3::GENIE3()'.

Details

This function infers GRNs with ARACNE, GENIE3 and CLR, ranks correlation weights for each GRN and calculates the average rank for each edge. Then, the resulting GRN is filtered to keep the top n edges that lead to the optimal scale-free topology fit.

Value

A filtered edge list with regulators in the first column and targets in the second column.

Examples

```
data(filt.se)
tfs <- sample(rownames(filt.se), size=50, replace=FALSE)
# Test with small number of trees for demonstration purpose
grn <- exp2grn(filt.se, regulators = tfs, nTrees=2, nsplit=2)
```

`exp_genes2orthogroups` *Collapse gene-level expression data to orthogroup level*

Description

For a given list of expression data, this function replaces genes with their corresponding orthogroups to allow inter-species comparisons.

Usage

```
exp_genes2orthogroups(explist = NULL, og = NULL, summarize = "median")
```

Arguments

`explist`	List of expression data frames or SummarizedExperiment objects.
`og`	Data frame of 3 columns corresponding to orthogroup, species ID, and gene ID, respectively. Species IDs must be the same as the names of the expression list.
`summarize`	Centrality measure to summarize multiple paralogous genes in the same orthogroup. One of "median" or "mean". Default: "median".

Value

List of expression data frames for each species with expression summarized at the orthogroup level.

Examples

```
data(og.zma.osa)
data(zma.se)
data(osa.se)
explist <- list(zma = zma.se,
                osa = osa.se)
og <- og.zma.osa
exp_ortho <- exp_genes2orthogroups(explist, og, summarize = "mean")
```

exp_preprocess *Preprocess expression data for network reconstruction*

Description

Preprocess expression data for network reconstruction

Usage

```
exp_preprocess(
  exp,
  NA_rm = TRUE,
  replaceby = 0,
  Zk_filtering = TRUE,
  zk = -2,
  cor_method = "spearman",
  remove_nonexpressed = TRUE,
  method = "median",
  min_exp = 1,
  min_percentage_samples = 0.25,
  remove_confounders = TRUE,
  variance_filter = FALSE,
  n = NULL,
  percentile = NULL,
  vstransform = FALSE
)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
`NA_rm`	Logical. It specifies whether to remove missing values from the expression data frame or not. Default = TRUE.
`replaceby`	If NA_rm is TRUE, what to use instead of NAs. One of 0 or ’mean’. Default is 0.
`Zk_filtering`	Logical. It specifies whether to filter outlying samples by Zk or not. Default: TRUE.
`zk`	If Zk_filtering is TRUE, the standardized connectivity threshold. Samples below this threshold will be considered outliers. Default is -2.
`cor_method`	If Zk_filtering is TRUE, the correlation method to use. One of ’spearman’, ’bicor’, or ’pearson’. Default is ’spearman’.
`remove_nonexpressed`	Logical. It specifies whether non-expressed genes should be removed or not. Default is TRUE.

`method`	If remove_nonexpressed is TRUE, the criterion to filter non-expressed genes out. One of "mean", "median", "percentage", or "allsamples". Default is 'median'.
`min_exp`	If method is 'mean', 'median', or 'allsamples', the minimum value for a gene to be considered expressed. If method is 'percentage', the minimum value each gene must have in at least n percent of samples to be considered expressed.
`min_percentage_samples`	If method is 'percentage', expressed genes must have expression >= min_exp in at least this percentage. Values must range from 0 to 1. Default = 0.25.
`remove_confounders`	Logical. If TRUE, it removes principal components that add noise to the data.
`variance_filter`	Logical. If TRUE, it will filter genes by variance. Default is FALSE.
`n`	If variance_filter is TRUE, the number of most variable genes to keep.
`percentile`	If variance_filter is TRUE, the percentage of most variable genes to keep.
`vstransform`	Logical indicating if data should be variance stabilizing transformed. This parameter can only be set to TRUE if data is a matrix of raw read counts.

Value

Processed gene expression data frame with gene IDs in row names and sample names in column names or ‘SummarizedExperiment‘ object.

Author(s)

Fabricio Almeida-Silva

References

Love, M. I., Huber, W., & Anders, S. (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome biology, 15(12), 1-21.

See Also

`varianceStabilizingTransformation`

Examples

```
data(zma.se)
exp <- exp_preprocess(zma.se, variance_filter=TRUE, n=1000)
```

`filt.se` *Filtered maize gene expression data from Shin et al., 2021.*

Description

Filtered expression data in transcripts per million (TPM) from Shin et al., 2021. This is the same data set described in `zma.se`, but it only contains the top 500 genes with the highest variances. This data set was created to be used in unit tests and examples.

Usage

```
data(filt.se)
```

Format

An object of class `SummarizedExperiment`

References

Shin, J., Marx, H., Richards, A., Vaneechoutte, D., Jayaraman, D., Maeda, J., ... & Roy, S. (2021). A network-based comparative framework to study conservation and divergence of proteomes in plant phylogenies. Nucleic Acids Research, 49(1), e3-e3.

Examples

```
data(filt.se)
```

`filter_by_variance` *Keep only genes with the highest variances*

Description

Keep only genes with the highest variances

Usage

```
filter_by_variance(exp, n = NULL, percentile = NULL)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
`n`	Number of most variable genes (e.g., n=5000 will keep the top 5000 most variable genes).
`percentile`	Percentile of most highly variable genes (e.g., percentile=0.1 will keep the top 10 percent most variable genes). Values must range from 0 to 1.

Value

Expression data frame or ‘SummarizedExperiment‘ object with the most variable genes in row names and samples in column names.

Author(s)

Fabricio Almeida-Silva

Examples

```
data(zma.se)
filt_exp <- filter_by_variance(zma.se, p=0.1)
```

gene_significance *Calculate gene significance for a given group of genes*

Description

Calculate gene significance for a given group of genes

Usage

```
gene_significance(
  exp,
  metadata,
  metadata_cols = NULL,
  genes = NULL,
  alpha = 0.05,
  cor_method = "pearson",
  min_cor = 0.2,
  use_abs = TRUE
)
```

Arguments

exp	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
metadata	A data frame containing sample names in row names and sample annotation in the first column. Ignored if ‘exp‘ is a ‘SummarizedExperiment‘ object, since the function will extract colData.
metadata_cols	A vector (either numeric or character) indicating which columns should be extracted from column metadata if **exp** is a ‘SummarizedExperiment‘ object. The vector can contain column indices (numeric) or column names (character). By default, all columns are used.
genes	Character vector of genes to be correlated with traits. If not given, all genes in ‘exp‘ will be considered.

`alpha`	Significance level. Default is 0.05.
`cor_method`	Method to calculate correlation. One of 'pearson', 'spearman' or 'kendall'. Default is 'spearman'.
`min_cor`	Minimum correlation coefficient. Default is 0.2.
`use_abs`	Logical indicating whether to filter by correlation using absolute value or not. If TRUE, a `min_cor` of say 0.2 would keep all correlations above 0.2 and below -0.2. Default is TRUE.

Value

A data frame with correlation and correlation p-values for each pair of gene and trait, with the following variables:

gene Factor, gene ID.

trait Factor, trait name. Each trait corresponds to a variable of the sample metadata (if numeric) or levels of a variable (if categorical).

cor Numeric, correlation.

pvalue Numeric, correlation P-values.

group Character, name of the metadata variable.

Author(s)

Fabricio Almeida-Silva

Examples

```
data(filt.se)
gs <- gene_significance(filt.se)
```

`get_edge_list` *Get edge list from an adjacency matrix for a group of genes*

Description

Get edge list from an adjacency matrix for a group of genes

Usage

```
get_edge_list(
  net,
  genes = NULL,
  module = NULL,
  filter = FALSE,
  method = "optimalSFT",
  r_optimal_test = seq(0.4, 0.9, by = 0.1),
  Zcutoff = 1.96,
  pvalue_cutoff = 0.05,
  rcutoff = 0.7,
  nSamples = NULL,
  check_SFT = FALSE,
  bp_param = BiocParallel::SerialParam()
)
```

Arguments

`net`	List object returned by `exp2gcn`.
`genes`	Character vector containing a subset of genes from which edges will be extracted. It can be ignored if the user wants to extract an edge list for a given module instead of individual genes.
`module`	Character with module name from which edges will be extracted. To include 2 or more modules, input the names in a character vector.
`filter`	Logical indicating whether to filter the edge list or not.
`method`	Method to filter spurious correlations. One of "Zscore", "optimalSFT", "pvalue" or "min_cor". See details for more information on the methods. Default: 'optimalSFT'
`r_optimal_test`	Numeric vector with the correlation thresholds to be tested for optimal scale-free topology fit. Only valid if `method` equals "optimalSFT". Default: seq(0.4, 0.9, by = 0.1)
`Zcutoff`	Minimum Z-score threshold. Only valid if `method` equals "Zscore". Default: 1.96
`pvalue_cutoff`	Maximum P-value threshold. Only valid if `method` equals "pvalue". Default: 0.05
`rcutoff`	Minimum correlation threshold. Only valid if `method` equals "min_cor". Default: 0.7
`nSamples`	Number of samples in the data set from which the correlation matrix was calculated. Only required if `method` equals "pvalue".
`check_SFT`	Logical indicating whether to test if the resulting network is close to a scale-free topology or not. Default: FALSE.
`bp_param`	BiocParallel back-end to be used. Default: BiocParallel::SerialParam()

Details

The default method ("optimalSFT") will create several different edge lists by filtering the original correlation matrix by the thresholds specified in `r_optimal_test`. Then, it will calculate a scale-free topology fit index for each of the possible networks and return the network that best fits the scale-free topology. The method "Zscore" will apply a Fisher Z-transformation for the correlation coefficients and remove the Z-scores below the threshold specified in `Zcutoff`. The method "pvalue" will calculate Student asymptotic p-value for the correlations and remove correlations whose p-values are above the threshold specified in `pvalue_cutoff`. The method "min_cor" will remove correlations below the minimum correlation threshold specified in `rcutoff`.

Value

Data frame with edge list for the input genes.

Author(s)

Fabricio Almeida-Silva

See Also

scaleFreeFitIndex

SFT_fit

exp2gcn.

Examples

```
data(filt.se)
gcn <- exp2gcn(filt.se, SFTpower = 18, cor_method = "pearson")
genes <- rownames(filt.se)[1:50]
edges <- get_edge_list(gcn, genes=genes, filter = FALSE)
```

`get_HK` *Get housekeeping genes from global expression profile*

Description

Get housekeeping genes from global expression profile

Usage

```
get_HK(exp)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.

Details

This function identifies housekeeping genes, which are broadly expressed genes with low variation in a global scale across samples. For some cases, users would want to remove these genes as they are not interesting for coexpression network analyses. See references for more details.

Value

Character vector of housekeeping gene IDs.

Author(s)

Fabricio Almeida-Silva

References

Machado, F.B., Moharana, K.C., Almeida-Silva, F., Gazara, R.K., Pedrosa-Silva, F., Coelho, F.S., Grativol, C. and Venancio, T.M. (2020), Systematic analysis of 1298 RNA-Seq samples and construction of a comprehensive soybean (Glycine max) expression atlas. Plant J, 103: 1894-1909.

Examples

```
data(zma.se)
hk <- get_HK(zma.se)
```

get_hubs_gcn *Get GCN hubs*

Description

Get GCN hubs

Usage

```
get_hubs_gcn(exp, net)
```

Arguments

exp	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
net	List object returned by exp2gcn.

Value

Data frame containing gene IDs, modules and intramodular connectivity of all hubs.

Author(s)

Fabricio Almeida-Silva

See Also

signedKME

Examples

```
data(filt.se)
gcn <- exp2gcn(filt.se, SFTpower = 18, cor_method = "pearson")
hubs <- get_hubs_gcn(filt.se, gcn)
```

`get_hubs_grn` *Get hubs for gene regulatory network*

Description

Get hubs for gene regulatory network

Get hubs for protein-protein interaction network

Usage

```
get_hubs_grn(
  edgelist,
  top_percentile = 0.1,
  top_n = NULL,
  return_degree = FALSE,
  ranked = TRUE
)

get_hubs_ppi(
  edgelist,
  top_percentile = 0.1,
  top_n = NULL,
  return_degree = FALSE
)
```

Arguments

`edgelist`	A protein-protein interaction network represented as an edge list.
`top_percentile`	Numeric from 0 to 1 indicating the percentage of proteins with the highest degree to consider hubs. Default: 0.1.
`top_n`	Numeric indicating the number of proteins with the highest degree to consider hubs.
`return_degree`	Logical indicating whether to return a data frame of degree for all proteins. If TRUE, the function will return a list instead of a data frame. Default: FALSE.
`ranked`	Logical indicating whether to treat third column of the edge list (edge weights) as ranked values. Ignored if the edge list only contains 2 columns. Default: TRUE.

Value

A data frame with gene ID in the first column and out degree in the second column or a list of two data frames with hubs and degree for all genes, respectively.

A data frame with protein ID in the first column and degree in the second column or a list of two data frames with hubs and degree for all genes, respectively.

Examples

```
data(filt.se)
tfs <- sample(rownames(filt.se), size=50, replace=FALSE)
grn_list <- grn_combined(filt.se, regulators=tfs, nTrees=2)
ranked_grn <- grn_average_rank(grn_list)
# split in only 2 groups for demonstration purposes
filtered_edges <- grn_filter(ranked_grn, nsplit=2)
hubs <- get_hubs_grn(filtered_edges)
ppi_edges <- igraph::get.edgelist(igraph::barabasi.game(n=500, directed=FALSE))
hubs <- get_hubs_ppi(ppi_edges, return_degree = TRUE)
```

---

get_neighbors *Get 1st-order neighbors of a given gene or group of genes*

Description

Get 1st-order neighbors of a given gene or group of genes

Usage

```
get_neighbors(genes, net, cor_threshold = 0.7)
```

Arguments

genes	Character vector containing genes from which direct neighbors will be extracted.
net	List object returned by exp2gcn.
cor_threshold	Correlation threshold to filter connections. As a weighted network is a fully connected graph, a cutoff must be selected. Default is 0.7.

Value

List containing 1st-order neighbors for each input gene.

Author(s)

Fabricio Almeida-Silva

See Also

exp2gcn SFT_fit

Examples

```
data(filt.se)
genes <- rownames(filt.se)[1:10]
gcn <- exp2gcn(filt.se, SFTpower = 18, cor_method = "pearson")
neighbors <- get_neighbors(genes, gcn)
```

grn_average_rank — *Rank edge weights for GRNs and calculate average across different methods*

Description

Rank edge weights for GRNs and calculate average across different methods

Usage

```
grn_average_rank(list_edges)
```

Arguments

`list_edges`	List containing edge lists as returned by the function `grn_combined`.

Value

Edge list containing regulator, target and mean rank from all algorithms.

Examples

```
data(filt.se)
tfs <- sample(rownames(filt.se), size=50, replace=FALSE)
grn_list <- grn_combined(filt.se, regulators=tfs, nTrees=2)
ranked_grn <- grn_average_rank(grn_list)
```

grn_combined — *Infer gene regulatory network with multiple algorithms and combine results in a list*

Description

Infer gene regulatory network with multiple algorithms and combine results in a list

Usage

```
grn_combined(
  exp,
  regulators = NULL,
  eps = 0.1,
  estimator_aracne = "spearman",
  estimator_clr = "pearson",
  remove_zero = TRUE,
  ...
)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
`regulators`	A character vector of regulators (e.g., transcription factors or miRNAs). All regulators must be included in ‘exp‘.
`eps`	Numeric value indicating the threshold used when removing an edge: for each triplet of nodes (i,j,k), the weakest edge, say (ij), is removed if its weight is below min(ik),(jk) - eps. Default: 0.1.
`estimator_aracne`	Entropy estimator to be used in ARACNE inference. One of "mi.empirical", "mi.mm", "mi.shrink", "mi.sg", "pearson", "spearman", or "kendall". Default: "spearman".
`estimator_clr`	Entropy estimator to be used in CLR inference. One of "mi.empirical", "mi.mm", "mi.shrink", "mi.sg", "pearson", "spearman", or "kendall". Default: "pearson".
`remove_zero`	Logical indicating whether to remove edges whose weight is exactly zero. Zero values indicate edges that were removed by ARACNE. Default: TRUE.
`...`	Additional arguments passed to ‘GENIE3::GENIE3()‘.

Value

A list of data frames representing edge lists. Each list element is an edge list for a specific method.

Examples

```
data(filt.se)
tfs <- sample(rownames(filt.se), size=50, replace=FALSE)
grn_list <- grn_combined(filt.se, regulators=tfs, nTrees=2)
```

`grn_filter` *Filter a gene regulatory network based on optimal scale-free topology fit*

Description

Filter a gene regulatory network based on optimal scale-free topology fit

Usage

```
grn_filter(edgelist, nsplit = 10, bp_param = BiocParallel::SerialParam())
```

Arguments

`edgelist`	A gene regulatory network represented as an edge list.
`nsplit`	Number of groups in which the edge list will be split. Default: 10.
`bp_param`	BiocParallel back-end to be used. Default: BiocParallel::SerialParam()

Details

The edge list will be split in n groups and the scale-free topology fit will be tested for each subset of the edge list. For instance, if an edge list of 10000 rows is used as input, the function will test SFT fit for the top 1000 edges, then top 2000 edges, and so on up to the whole edge list.

Value

The edge list that best fits the scale-free topology.

Examples

```
data(filt.se)
tfs <- sample(rownames(filt.se), size=50, replace=FALSE)
grn_list <- grn_combined(filt.se, regulators=tfs, nTrees=2)
ranked_grn <- grn_average_rank(grn_list)
# split in only 2 groups for demonstration purposes
filtered_edges <- grn_filter(ranked_grn, nsplit=2)
```

`grn_infer` *Infer gene regulatory network with one of three algorithms*

Description

The available algorithms are Context Likelihood of Relatedness (CLR), ARACNE, or GENIE3.

Usage

```
grn_infer(
  exp,
  regulators = NULL,
  method = c("clr", "aracne", "genie3"),
  estimator_clr = "pearson",
  estimator_aracne = "spearman",
  eps = 0.1,
  remove_zero = TRUE,
  ...
)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
`regulators`	A character vector of regulators (e.g., transcription factors or miRNAs). All regulators must be included in ‘exp‘.
`method`	GRN inference algorithm to be used. One of "clr", "aracne", or "genie3".
`estimator_clr`	Entropy estimator to be used. One of "mi.empirical", "mi.mm", "mi.shrink", "mi.sg", "pearson", "spearman", or "kendall". Default: "pearson".

`estimator_aracne`	Entropy estimator to be used. One of "mi.empirical", "mi.mm", "mi.shrink", "mi.sg", "pearson", "spearman", or "kendall". Default: "spearman".
`eps`	Numeric value indicating the threshold used when removing an edge: for each triplet of nodes (i,j,k), the weakest edge, say (ij), is removed if its weight is below min(ik),(jk) - eps. Default: 0.1.
`remove_zero`	Logical indicating whether to remove edges whose weight is exactly zero. Default: TRUE
`...`	Additional arguments passed to 'GENIE3::GENIE3()'.

Value

A gene regulatory network represented as an edge list.

Examples

```
data(filt.se)
tfs <- sample(rownames(filt.se), size=20, replace=FALSE)
clr <- grn_infer(filt.se, method = "clr", regulators=tfs)
aracne <- grn_infer(filt.se, method = "aracne", regulators=tfs)
# only 2 trees for demonstration purposes
genie3 <- grn_infer(filt.se, method = "genie3", regulators=tfs, nTrees=2)
```

`is_singleton` *Logical expression to check if gene or gene set is singleton or not*

Description

Logical expression to check if gene or gene set is singleton or not

Usage

```
is_singleton(genes, og)
```

Arguments

`genes`	Character containing gene or group of genes to be evaluated.
`og`	Data frame of 3 columns corresponding to orthogroup, species ID, and gene ID, respectively.

Value

Vector of logical values indicating if gene or group of genes is singleton or not.

Author(s)

Fabricio Almeida-Silva

See Also

`is_duplicated`

Examples

```
data(og.zma.osa)
data(filt.se)
genes <- tail(rownames(filt.se), n = 100)
is_singleton(genes, og.zma.osa)
```

modPres_netrep — *Calculate module preservation between two expression data sets using NetRep's algorithm*

Description

Calculate module preservation between two expression data sets using NetRep's algorithm

Usage

```
modPres_netrep(
  explist,
  ref_net = NULL,
  test_net = NULL,
  nPerm = 1000,
  nThreads = 1
)
```

Arguments

`explist`	List of expression data frames or SummarizedExperiment objects.
`ref_net`	Reference network object returned by the function `exp2net`.
`test_net`	Test network object returned by the function `exp2net`.
`nPerm`	Number of permutations. Default: 1000
`nThreads`	Number of threads to be used for parallel computing. Default: 1

Value

Output list from `NetRep::modulePreservation` and a message in user's standard output stating which modules are preserved.

See Also

`modulePreservation`

Examples

```
set.seed(1)
data(og.zma.osa)
data(zma.se)
data(osa.se)
og <- og.zma.osa
exp_ortho <- exp_genes2orthogroups(explist, og, summarize = "mean")
exp_ortho <- lapply(exp_ortho, function(x) filter_by_variance(x, n=1500))
# Previously calculated SFT powers
powers <- c(13, 15)
gcn_osa <- exp2gcn(exp_ortho$osa, net_type = "signed hybrid",
                   SFTpower = powers[1], cor_method = "pearson")
gcn_zma <- exp2gcn(exp_ortho$zma, net_type = "signed hybrid",
                   SFTpower = powers[2], cor_method = "pearson")
explist <- exp_ortho
ref_net <- gcn_osa
test_net <- gcn_zma
# 10 permutations for demonstration purposes
pres_netrep <- modPres_netrep(explist, ref_net, test_net,
                              nPerm=10, nThreads = 2)
```

`modPres_WGCNA` *Calculate module preservation between two expression data sets using WGCNA's algorithm*

Description

Calculate module preservation between two expression data sets using WGCNA's algorithm

Usage

```
modPres_WGCNA(explist, ref_net, nPerm = 200)
```

Arguments

`explist`	List of expression data frames or SummarizedExperiment objects.
`ref_net`	Reference network object returned by the function `exp2net`.
`nPerm`	Number of permutations for the module preservation statistics. It must be greater than 1. Default: 200.

Value

A ggplot object with module preservation statistics.

Examples

```
set.seed(1)
data(og.zma.osa)
data(zma.se)
data(osa.se)
explist <- list(Zma = zma.se, Osa = osa.se)
og <- og.zma.osa
exp_ortho <- exp_genes2orthogroups(explist, og, summarize = "mean")
exp_ortho <- lapply(exp_ortho, function(x) filter_by_variance(x, n=1500))
# Previously calculated power
powers <- c(13, 15)
gcn_osa <- exp2gcn(exp_ortho$Osa, net_type = "signed hybrid",
                   SFTpower = powers[1], cor_method = "pearson")
explist <- exp_ortho
ref_net <- gcn_osa
# 5 permutations for demonstration purposes
pres_wgcna <- modPres_WGCNA(explist, ref_net, nPerm=5)
```

`module_enrichment` *Perform enrichment analysis for coexpression network modules*

Description

Perform enrichment analysis for coexpression network modules

Usage

```
module_enrichment(
  net = NULL,
  background_genes,
  annotation,
  column = NULL,
  correction = "BH",
  p = 0.05,
  min_setsize = 10,
  max_setsize = 500,
  bp_param = BiocParallel::SerialParam()
)
```

Arguments

`net`	List object returned by `exp2gcn`.
`background_genes`	Character vector of genes to be used as background for the Fisher's Exact Test.
`annotation`	Annotation data frame with genes in the first column and functional annotation in the other columns. This data frame can be exported from Biomart or similar databases.

`column`	Column or columns of `annotation` to be used for enrichment. Both character or numeric values with column indices can be used. If users want to supply more than one column, input a character or numeric vector. Default: all columns from `annotation`.
`correction`	Multiple testing correction method. One of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr" or "none". Default is "BH".
`p`	P-value threshold. P-values below this threshold will be considered significant. Default is 0.05.
`min_setsize`	Numeric indicating the minimum gene set size to be considered. Gene sets correspond to levels of each variable in **annotation**). Default: 10.
`max_setsize`	Numeric indicating the maximum gene set size to be considered. Gene sets correspond to levels of each variable in **annotation**). Default: 500.
`bp_param`	BiocParallel back-end to be used. Default: BiocParallel::SerialParam()

Value

A data frame of overrepresentation results with the following variables:

term character, functional term ID/name.

genes numeric, intersection length between input genes and genes in a particular functional term.

all numeric, number of all genes in a particular functional term.

pval numeric, P-value for the hypergeometric test.

padj numeric, P-value adjusted for multiple comparisons using the method specified in parameter **adj**.

category character, name of the grouping variable (i.e., column name of **annotation**).

module character, module name.

Author(s)

Fabricio Almeida-Silva

Examples

```
data(filt.se)
data(zma.interpro)
background <- rownames(filt.se)
gcn <- exp2gcn(filt.se, SFTpower = 18, cor_method = "pearson")
mod_enrich <- module_enrichment(gcn, background, zma.interpro, p=1)
```

module_preservation *Calculate network preservation between two expression data sets*

Description

Calculate network preservation between two expression data sets

Usage

```
module_preservation(
  explist,
  ref_net = NULL,
  test_net = NULL,
  algorithm = "netrep",
  nPerm = 1000,
  nThreads = 1
)
```

Arguments

explist	List of SummarizedExperiment objects or expression data frames with genes (or orthogroups) in row names and samples in column names.
ref_net	Reference network object returned by the function `exp2gcn`.
test_net	Test network object returned by the function `exp2gcn`.
algorithm	Module preservation algorithm to be used. One of 'netrep' (default, permutation-based) or WGCNA.
nPerm	Number of permutations. Default: 1000
nThreads	Number of threads to be used for parallel computing. Default: 1

Value

A list containing the preservation statistics (netrep) or a ggplot object with preservation statistics. See `WGCNA::modulePreservation` or `NetRep::modulePreservation` for more info.

Examples

```
set.seed(1)
data(og.zma.osa)
data(zma.se)
data(osa.se)
og <- og.zma.osa
exp_ortho <- exp_genes2orthogroups(explist, og, summarize = "mean")
exp_ortho <- lapply(exp_ortho, function(x) filter_by_variance(x, n=1500))
# Previously calculated SFT powers
powers <- c(13, 15)
gcn_osa <- exp2gcn(exp_ortho$osa, net_type = "signed hybrid",
                   SFTpower = powers[1], cor_method = "pearson")
```

```
gcn_zma <- exp2gcn(exp_ortho$zma, net_type = "signed hybrid",
                   SFTpower = powers[2], cor_method = "pearson")
explist <- exp_ortho
ref_net <- gcn_osa
test_net <- gcn_zma
# 10 permutations for demonstration purposes
pres <- module_preservation(explist, ref_net, test_net, nPerm=10)
```

`module_stability` *Perform module stability analysis*

Description

Perform module stability analysis

Usage

```
module_stability(exp, net, nRuns = 20)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a 'SummarizedExperiment' object.
`net`	List object returned by `exp2gcn`.
`nRuns`	Number of times to resample. Default is 20.

Value

A base plot with the module stability results.

See Also

`sampledBlockwiseModules`

Examples

```
data(filt.se)
filt <- filt.se[1:100, ] # reducing even further for testing purposes
# The SFT fit was previously calculated and the optimal power was 16
gcn <- exp2gcn(filt, SFTpower = 16, cor_method = "pearson")
# For simplicity, only 2 runs
module_stability(exp = filt, net = gcn, nRuns = 2)
```

`module_trait_cor` *Correlate module eigengenes to trait*

Description

Correlate module eigengenes to trait

Usage

```
module_trait_cor(
  exp,
  metadata,
  MEs,
  metadata_cols = NULL,
  cor_method = "pearson"
)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
`metadata`	A data frame containing sample names in row names and sample annotation in the first column. Ignored if ‘exp‘ is a ‘SummarizedExperiment‘ object, since the function will extract colData.
`MEs`	Module eigengenes. It is the 2nd element of the result list generated by the function `exp2gcn`.
`metadata_cols`	A vector (either numeric or character) indicating which columns should be extracted from column metadata if **exp** is a ‘SummarizedExperiment‘ object. The vector can contain column indices (numeric) or column names (character). By default, all columns are used.
`cor_method`	Method to calculate correlation. One of ’pearson’, ’spearman’ or ’kendall’. Default is ’spearman’.

Value

A data frame with correlation and correlation p-values for each pair of ME and trait, with the following variables:

ME Factor, module eigengene.

trait Factor, trait name. Each trait corresponds to a variable of the sample metadata (if numeric) or levels of a variable (if categorical).

cor Numeric, correlation.

pvalue Numeric, correlation P-values.

group Character, name of the metadata variable.

Author(s)

Fabricio Almeida-Silva

Examples

```
data(filt.se)
gcn <- exp2gcn(filt.se, SFTpower = 18, cor_method = "pearson")
module_trait_cor(filt.se, MEs = gcn$MEs)
```

`net_stats` *Calculate network statistics*

Description

Calculate network statistics

Usage

```
net_stats(
  adj_matrix = NULL,
  net_type = c("gcn", "ppi", "grn"),
  calculate_additional = FALSE
)
```

Arguments

`adj_matrix`	Adjacency matrix that represents the network.
`net_type`	One of "gcn" (gene coexpression network), "ppi" (protein-protein interaction), or "grn" (gene regulatory network).
`calculate_additional`	Logical indicating whether to calculate additional network statistics (betweenness and closeness). Default is FALSE.

Value

A list containing the following elements:

- Connectivity
- ScaledConnectivity
- ClusterCoef
- MAR (for gcn only)
- Density
- Centralization
- Heterogeneity (gcn only)
- Diameter
- Betweenness
- Closeness

See Also

graph_from_adjacency_matrix, cliques,diameter, estimate_betweenness,V, closeness,degree, transitivity,edge_density, centr_degree fundamentalNetworkConcepts

Examples

```
data(filt.se)
set.seed(12)
filt.se <- exp_preprocess(
    filt.se, Zk_filtering = FALSE, variance_filter = TRUE, n = 200
)
gcn <- exp2gcn(
    filt.se, SFTpower = 7, cor_method = "pearson", net_type = "signed hybrid"
)
stats <- net_stats(gcn$adjacency_matrix, net_type = "gcn")
```

---

og.zma.osa *Orthogroups between maize and rice*

Description

The orthogroups were downloaded from the PLAZA 4.0 Monocots database.

Usage

```
data(og.zma.osa)
```

Format

A 3-column data frame with orthogroups, species IDs and gene IDs.

References

Van Bel, M., Diels, T., Vancaester, E., Kreft, L., Botzki, A., Van de Peer, Y., ... & Vandepoele, K. (2018). PLAZA 4.0: an integrative resource for functional, evolutionary and comparative plant genomics. Nucleic acids research, 46(D1), D1190-D1196.

Examples

```
data(og.zma.osa)
```

`osa.se` *Rice gene expression data from Shin et al., 2021.*

Description

Filtered expression data in transcripts per million (TPM) from Shin et al., 2021. Genes with TPM values <5 in more than 60 were removed to reduce package size. The expression data and associated sample metadata are stored in a SummarizedExperiment object.

Usage

```
data(osa.se)
```

Format

An object of class `SummarizedExperiment`

References

Shin, J., Marx, H., Richards, A., Vaneechoutte, D., Jayaraman, D., Maeda, J., ... & Roy, S. (2021). A network-based comparative framework to study conservation and divergence of proteomes in plant phylogenies. Nucleic Acids Research, 49(1), e3-e3.

Examples

```
data(osa.se)
```

`parse_orthofinder` *Parse orthogroups identified by OrthoFinder*

Description

This function converts the orthogroups file named **Orthogroups.tsv** to a 3-column data frame that can be interpreted by BioNERO.

Usage

```
parse_orthofinder(file_path = NULL)
```

Arguments

`file_path`	Path to Orthogroups/Orthogroups.tsv file generated by OrthoFinder.

Value

A 3-column data frame with orthogroups, species IDs and gene IDs, respectively.

Author(s)

Fabricio Almeida-Silva

Examples

```
path <- system.file("extdata", "Orthogroups.tsv", package = "BioNERO")
og <- parse_orthofinder(path)
```

`PC_correction` *Apply Principal Component (PC)-based correction for confounding artifacts*

Description

Apply Principal Component (PC)-based correction for confounding artifacts

Usage

```
PC_correction(exp, verbose = FALSE)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
`verbose`	Logical indicating whether to display progress messages or not. Default: FALSE.

Value

Corrected expression data frame or ‘SummarizedExperiment‘ object.

Author(s)

Fabricio Almeida-Silva

References

Parsana, P., Ruberman, C., Jaffe, A. E., Schatz, M. C., Battle, A., & Leek, J. T. (2019). Addressing confounding artifacts in reconstruction of gene co-expression networks. Genome biology, 20(1), 1-6.

See Also

`num.sv`,`sva_network`

Examples

```
data(zma.se)
exp <- filter_by_variance(zma.se, n=500)
exp <- PC_correction(exp)
```

`plot_dendro_and_colors` *Plot dendrogram of genes and modules*

Description

Plot dendrogram of genes and modules

Usage

```
plot_dendro_and_colors(gcn)
```

Arguments

`gcn`	List object returned by `exp2gcn`.

Value

A base plot with the gene dendrogram and modules.

Examples

```
data(filt.se)
gcn <- exp2gcn(filt.se, SFTpower = 18, cor_method = "pearson")
plot_dendro_and_colors(gcn)
```

`plot_eigengene_network` *Plot eigengene network*

Description

Plot eigengene network

Usage

```
plot_eigengene_network(gcn, palette = "PRGn")
```

Arguments

`gcn`	List object returned by `exp2gcn`.
`palette`	Character indicating the name of the RColorBrewer palette to use. Default: "PRGn".

Value

A base plot with the eigengene network

Examples

```
data(filt.se)
gcn <- exp2gcn(filt.se, SFTpower = 18, cor_method = "pearson")
plot_eigengene_network(gcn)
```

plot_expression_profile

Plot expression profile of given genes across samples

Description

Plot expression profile of given genes across samples

Usage

```
plot_expression_profile(
  genes,
  exp,
  metadata,
  metadata_cols = 1,
  plot_module = TRUE,
  net,
  modulename,
  bg_line = "mean"
)
```

Arguments

`genes`	Character vector containing a subset of genes from which edges will be extracted. It can be ignored if `plot_module` is TRUE.
`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
`metadata`	A data frame of sample metadata containing sample names in row names and sample annotation in subsequent columns. Ignored if ‘exp‘ is a ‘SummarizedExperiment‘ object, since colData will be automatically extracted.
`metadata_cols`	A character or numeric scalar indicating which column should be extracted from column metadata if **exp** is a ‘SummarizedExperiment‘ object. The column to be extracted can be represented by indices (numeric) or column names (character). By default, the first column is used.
`plot_module`	Logical indicating whether to plot a whole module or not. If set to FALSE, `genes` must be specified.
`net`	List object returned by `exp2gcn`.
`modulename`	Name of the module to plot.
`bg_line`	Character indicating what to show in the background (black) line. One of "mean" or "median". Default: "mean".

Value

A ggplot object showing the expression profile of some genes across all samples.

Author(s)

Fabricio Almeida-Silva

Examples

```
data(zma.se)
data(filt.se)
genes <- rownames(filt.se)
plot_expression_profile(genes = genes, exp = zma.se, plot_module = FALSE)
```

`plot_gcn` *Plot gene coexpression network from edge list*

Description

Plot gene coexpression network from edge list

Usage

```
plot_gcn(
  edgelist_gcn,
  net,
  color_by = "module",
  hubs = NULL,
  show_labels = "tophubs",
  top_n_hubs = 5,
  curvature = 0,
  interactive = FALSE,
  dim_interactive = c(600, 600)
)
```

Arguments

`edgelist_gcn`	Data frame containing the edge list for the GCN. The edge list can be generated with `get_edge_list()`.
`net`	List object returned by `exp2net`.
`color_by`	How should nodes be colored? It must be either "module" (nodes will have the colors of their modules) or a 2-column data frame containing genes in the first column and a custom gene annotation in the second column. Default: "module".
`hubs`	Data frame containing hub genes in the first column, their modules in the second column, and intramodular connectivity in the third column.

`show_labels`	Character indicating which nodes will be labeled. One of "all", "allhubs", "tophubs", or "none". Default: tophubs.
`top_n_hubs`	Number of top hubs to be labeled. It is only valid if `show_labels` equals "tophubs". Default is 5.
`curvature`	Numeric indicating the amount of curvature in edges. Negative values produce left-hand curves, positive values produce right-hand curves, and zero produces a straight line. Default: 0.1.
`interactive`	Logical indicating whether the network should be interactive or not. Default is FALSE.
`dim_interactive`	Numeric vector with width and height of window for interactive plotting. Default: c(600,600).

Value

A ggplot object.

Author(s)

Fabricio Almeida-Silva

Examples

```
data(filt.se)
gcn <- exp2gcn(filt.se, SFTpower = 18, cor_method = "pearson")
gcn_edges <- get_edge_list(gcn, module="brown", filter=TRUE,
                           method="min_cor")
hubs <- get_hubs_gcn(filt.se, gcn)
p <- plot_gcn(gcn_edges, gcn, hubs = hubs)
```

`plot_gene_significance`
Plot a heatmap of gene significance

Description

Plot a heatmap of gene significance

Usage

```
plot_gene_significance(corandp, palette = "RdYlBu", transpose = FALSE, ...)
```

Arguments

`corandp`	A data frame of gene-trait correlations as returned by `gene_significance()`.
`palette`	Character indicating which RColorBrewer palette to use. Default: 'RdYlBu'.
`transpose`	Logical indicating whether to transpose the heatmap or not.
`...`	Additional arguments to `ComplexHeatmap::pheatmap()`.

Details

Significance levels: 1 asterisk: significant at alpha = 0.05. 2 asterisks: significant at alpha = 0.01. 3 asterisks: significant at alpha = 0.001. no asterisk: not significant.

Value

A ‘Heatmap‘ object created by `ComplexHeatmap::pheatmap()`.

Examples

```
data(filt.se)
gcn <- exp2gcn(filt.se, SFTpower = 18, cor_method = "pearson")
corandp <- gene_significance(filt.se)
plot_gene_significance(corandp, show_rownames = FALSE)
```

`plot_grn` *Plot gene regulatory network from edge list*

Description

Plot gene regulatory network from edge list

Usage

```
plot_grn(
  edgelist_grn,
  show_labels = "tophubs",
  top_n_hubs = 5,
  layout = igraph::with_kk,
  arrow.gap = 0.01,
  ranked = TRUE,
  curvature = 0.1,
  interactive = FALSE,
  dim_interactive = c(600, 600)
)
```

Arguments

`edgelist_grn`	Data frame containing the edge list for the GRN network. First column is the TF and second column is the target gene. All other columns are interpreted as edge attributes.
`show_labels`	Character indicating which nodes will be labeled. One of "all", "allhubs", "tophubs", or "none".
`top_n_hubs`	Number of top hubs to be labeled. It is only valid if `show_labels` equals "tophubs". Default is 5.
`layout`	igraph function for the network layout. One of with_dh, with_drl, with_gem, with_lgl, with_fr, with_graphopt, with_kk and with_mds. Default is with_kk.

`arrow.gap`	Numeric indicating the distance between nodes and arrows. Default is 0.2.
`ranked`	Logical indicating whether to treat third column of the edge list (edge weights) as ranked values. Default: TRUE.
`curvature`	Numeric indicating the amount of curvature in edges. Negative values produce left-hand curves, positive values produce right-hand curves, and zero produces a straight line. Default: 0.1.
`interactive`	Logical indicating whether the network should be interactive or not. Default is FALSE.
`dim_interactive`	Numeric vector with width and height of window for interactive plotting. Default: c(600,600).

Value

A ggplot object containing the network.

Author(s)

Fabricio Almeida-Silva

Examples

```
data(filt.se)
tfs <- sample(rownames(filt.se), size = 50, replace = FALSE)
grn_edges <- grn_infer(filt.se, method = "clr", regulators = tfs)
p <- plot_grn(grn_edges, ranked = FALSE)
```

`plot_heatmap` *Plot heatmap of hierarchically clustered sample correlations or gene expression*

Description

Plot heatmap of hierarchically clustered sample correlations or gene expression

Usage

```
plot_heatmap(
  exp,
  col_metadata = NA,
  row_metadata = NA,
  coldata_cols = NULL,
  rowdata_cols = NULL,
  type = "samplecor",
  cor_method = "spearman",
  palette = NULL,
  log_trans = FALSE,
  ...
)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
`col_metadata`	A data frame containing sample names in row names and sample annotation in the subsequent columns. The maximum number of columns is 3 to ensure legends can be visualized. Ignored if ‘exp‘ is a ‘SummarizedExperiment‘ object, since the function will extract colData. Default: NA.
`row_metadata`	A data frame containing gene IDs in row names and gene functional classification in the first column. The maximum number of columns is 3 to ensure legends can be visualized. Default: NA.
`coldata_cols`	A vector (either numeric or character) indicating which columns should be extracted from column metadata if **exp** is a ‘SummarizedExperiment‘ object. The vector can contain column indices (numeric) or column names (character). By default, all columns are used.
`rowdata_cols`	A vector (either numeric or character) indicating which columns should be extracted from row metadata if **exp** is a ‘SummarizedExperiment‘ object. The vector can contain column indices (numeric) or column names (character). By default, all columns are used.
`type`	Type of heatmap to plot. One of ’samplecor’ (sample correlations) or ’expr’. Default: ’samplecor’.
`cor_method`	Correlation method to use in case **type** is "samplecor". One of ’spearman’ or ’pearson’. Default is ’spearman’.
`palette`	RColorBrewer palette to use. Default is "Blues" for sample correlation heatmaps and "YlOrRd" for gene expression heatmaps.
`log_trans`	Logical indicating whether to log transform the expression data or not. Default: FALSE.
`...`	Additional arguments to be passed to `ComplexHeatmap::pheatmap()`. These arguments can be used to control heatmap aesthetics, such as show/hide row and column names, change font size, activate/deactivate hierarchical clustering, etc. For a complete list of the options, see `?ComplexHeatmap::pheatmap()`.

Value

A heatmap of sample correlations or gene expression.

Author(s)

Fabricio Almeida-Silva

See Also

`RColorBrewer`

Examples

```
data(filt.se)
plot_heatmap(filt.se)
```

`plot_module_trait_cor` *Plot a heatmap of module-trait correlations*

Description

Plot a heatmap of module-trait correlations

Usage

```
plot_module_trait_cor(corandp, palette = "RdYlBu", transpose = FALSE)
```

Arguments

`corandp`	A data frame of module-trait correlations as returned by `module_trait_cor()`.
`palette`	Character indicating which RColorBrewer palette to use. Default: 'RdYlBu'.
`transpose`	Logical indicating whether to transpose the heatmap or not.

Details

Significance levels: 1 asterisk: significant at alpha = 0.05. 2 asterisks: significant at alpha = 0.01. 3 asterisks: significant at alpha = 0.001. no asterisk: not significant.

Value

A ‘Heatmap‘ object created by `ComplexHeatmap::pheatmap()`.

Examples

```
data(filt.se)
gcn <- exp2gcn(filt.se, SFTpower = 18, cor_method = "pearson")
corandp <- module_trait_cor(filt.se, MEs = gcn$MEs)
plot_module_trait_cor(corandp)
```

`plot_ngenes_per_module` *Plot number of genes per module*

Description

Plot number of genes per module

Usage

```
plot_ngenes_per_module(net = NULL)
```

Arguments

`net`	List object returned by `exp2gcn`.

Value

A ggplot object with a bar plot of gene number in each module.

Examples

```
data(filt.se)
gcn <- exp2gcn(filt.se, SFTpower = 18, cor_method = "pearson")
plot_ngenes_per_module(gcn)
```

`plot_PCA` *Plot Principal Component Analysis (PCA) of samples*

Description

Plot Principal Component Analysis (PCA) of samples

Usage

```
plot_PCA(
  exp,
  metadata,
  metadata_cols = NULL,
  log_trans = FALSE,
  PCs = c(1, 2),
  size = 2
)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
`metadata`	A data frame of sample metadata containing sample names in row names and sample annotation in subsequent columns. Ignored if ‘exp‘ is a ‘Summarized-Experiment‘ object, since colData will be automatically extracted.
`metadata_cols`	A vector (either numeric or character) indicating which columns should be extracted from column metadata if **exp** is a ‘SummarizedExperiment‘ object. The vector can contain column indices (numeric) or column names (character). By default, all columns are used.
`log_trans`	Logical indicating whether the gene expression matrix should be log transformed using `log(exp + 1)`. Default: FALSE.
`PCs`	Numeric vector of length 2 indicating the principal components to be plotted on the x-axis and y-axis, respectively. Default: `c(1, 2)`.
`size`	Numeric indicating the point size. Default is 2.

Value

A ggplot object with the PCA plot.

Author(s)

Fabricio Almeida-Silva

See Also

`ggplot`

Examples

```
data(zma.se)
plot_PCA(zma.se, log_trans = TRUE)
```

`plot_ppi` *Plot protein-protein interaction network from edge list*

Description

Plot protein-protein interaction network from edge list

Usage

```
plot_ppi(
  edgelist_int,
  color_by = "community",
  clustering_method = igraph::cluster_infomap,
  show_labels = "tophubs",
  top_n_hubs = 5,
  add_color_legend = TRUE,
  curvature = 0,
  interactive = FALSE,
  dim_interactive = c(600, 600)
)
```

Arguments

`edgelist_int`	Data frame containing the edge list for the PPI network. First column is the protein 1 and second column is the protein 2. All other columns are interpreted as edge attributes.
`color_by`	How should nodes be colored? It must be either "community" or a 2-column data frame containing proteins in the first column and a custom annotation in the second column. If "community", a clustering algorithm will be applied. Default: "community".
`clustering_method`	igraph function to be used for community detection. Available functions are cluster_infomap, cluster_edge_betweenness, cluster_fast_greedy, cluster_walktrap, cluster_spinglass, cluster_leading_eigen, cluster_louvain, and cluster_label_prop. Default is cluster_infomap.
`show_labels`	Character indicating which nodes will be labeled. One of "all", "allhubs", "tophubs", or "none".
`top_n_hubs`	Number of top hubs to be labeled. It is only valid if `show_labels` equals "tophubs". Default is 5.
`add_color_legend`	Logical indicating whether to add a color legend for nodes. Default: TRUE.
`curvature`	Numeric indicating the amount of curvature in edges. Negative values produce left-hand curves, positive values produce right-hand curves, and zero produces a straight line. Default: 0.
`interactive`	Logical indicating whether the network should be interactive or not. Default is FALSE.
`dim_interactive`	Numeric vector with width and height of window for interactive plotting. Default: c(600,600).

Value

A ggplot object.

Author(s)

Fabricio Almeida-Silva

Examples

```
ppi_edges <- igraph::get.edgelist(igraph::barabasi.game(n=50, directed=FALSE))
p <- plot_ppi(ppi_edges, add_color_legend = FALSE)
```

q_normalize *Quantile normalize the expression data*

Description

Quantile normalize the expression data

Usage

```
q_normalize(exp)
```

Arguments

exp	A gene expression data frame with genes in row names and samples in column names.

Value

Expression matrix with normalized values

Examples

```
data(zma.se)
exp <- SummarizedExperiment::assay(zma.se)
norm_exp <- q_normalize(exp)
```

remove_nonexp *Remove genes that are not expressed based on a user-defined threshold*

Description

Remove genes that are not expressed based on a user-defined threshold

Usage

```
remove_nonexp(
  exp,
  method = "median",
  min_exp = 1,
  min_percentage_samples = 0.25
)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
`method`	Criterion to filter non-expressed genes out. One of "mean", "median", "percentage", or "allsamples". Default is "median".
`min_exp`	If method is ’mean’, ’median’, or ’allsamples’, the minimum value for a gene to be considered expressed. If method is ’percentage’, the minimum value each gene must have in at least n percent of samples to be considered expressed.
`min_percentage_samples`	In case the user chooses ’percentage’ as method, expressed genes must have expression >= min_exp in at least this percentage. Values must range from 0 to 1.

Value

Filtered gene expression data frame or ‘SummarizedExperiment‘ object.

Author(s)

Fabricio Almeida-Silva

See Also

`rowMedians` `goodSamplesGenes`

Examples

```
data(zma.se)
filt_exp <- remove_nonexp(zma.se, min_exp = 5)
```

`replace_na` *Remove missing values in a gene expression data frame*

Description

Remove missing values in a gene expression data frame

Usage

```
replace_na(exp, replaceby = 0)
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
`replaceby`	What to use instead of NAs. One of 0 or ’mean’. Default is 0.

Value

Gene expression data frame or ‘SummarizedExperiment‘ object with all NAs replaced according to the argument ’replaceby’.

Author(s)

Fabricio Almeida-Silva

Examples

```
data(zma.se)
exp <- replace_na(zma.se)
sum(is.na(exp))
```

`SFT_fit` *Pick power to fit network to a scale-free topology*

Description

Pick power to fit network to a scale-free topology

Usage

```
SFT_fit(exp, net_type = "signed", rsquared = 0.8, cor_method = "spearman")
```

Arguments

`exp`	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
`net_type`	Network type. One of ’signed’, ’signed hybrid’ or ’unsigned’. Default is signed.
`rsquared`	R squared cutoff. Default is 0.8.
`cor_method`	Correlation method. One of "pearson", "biweight" or "spearman". Default is "spearman".

Value

A list containing:

- powerOptimal power based on scale-free topology fit
- plotA ggplot object displaying main statistics of the SFT fit test

Author(s)

Fabricio Almeida-Silva

See Also

pickSoftThreshold

Examples

```
data(filt.se)
sft <- SFT_fit(filt.se, cor_method = "pearson")
```

`ZKfiltering` *Filter outlying samples based on the standardized connectivity (Zk) method*

Description

Filter outlying samples based on the standardized connectivity (Zk) method

Usage

```
ZKfiltering(exp, zk = -2, cor_method = "spearman")
```

Arguments

exp	A gene expression data frame with genes in row names and samples in column names or a ‘SummarizedExperiment‘ object.
zk	Standardized connectivity threshold. Default is -2.
cor_method	Correlation method. One of "pearson", "biweight" or "spearman". Default is "spearman".

Value

Filtered gene expression data frame or ‘SummarizedExperiment‘ object.

Author(s)

Fabricio Almeida-Silva

References

Oldham, M. C., Langfelder, P., & Horvath, S. (2012). Network methods for describing sample relationships in genomic datasets: application to Huntington's disease. BMC systems biology, 6(1), 1-18.

See Also

`adjacency`

Examples

```
data(zma.se)
filt_exp <- ZKfiltering(zma.se)
```

`zma.interpro` *Maize Interpro annotation*

Description

Interpro protein domain annotation retrieved from the PLAZA Monocots 4.0 database. Only genes included in `zma.se` are present in this subset.

Usage

```
data(zma.interpro)
```

Format

A 2-column data frame containing gene IDs and their associated Interpro annotations.

References

Van Bel, M., Diels, T., Vancaester, E., Kreft, L., Botzki, A., Van de Peer, Y., ... & Vandepoele, K. (2018). PLAZA 4.0: an integrative resource for functional, evolutionary and comparative plant genomics. Nucleic acids research, 46(D1), D1190-D1196.

Examples

```
data(zma.interpro)
```

zma.se — *Maize gene expression data from Shin et al., 2021.*

Description

Filtered expression data in transcripts per million (TPM) from Shin et al., 2021. Genes with TPM values <5 in more than 60 were removed to reduce package size. The expression data and associated sample metadata are stored in a SummarizedExperiment object.

Usage

```
data(zma.se)
```

Format

An object of class `SummarizedExperiment`

References

Shin, J., Marx, H., Richards, A., Vaneechoutte, D., Jayaraman, D., Maeda, J., ... & Roy, S. (2021). A network-based comparative framework to study conservation and divergence of proteomes in plant phylogenies. Nucleic Acids Research, 49(1), e3-e3.

Examples

```
data(zma.se)
```

zma.tfs — *Maize transcription factors*

Description

Transcription factors and their families were downloaded from PlantTFDB 4.0.

Usage

```
data(zma.tfs)
```

Format

A data frame with gene IDs of TFs and their associated families.

References

Jin, J., Tian, F., Yang, D. C., Meng, Y. Q., Kong, L., Luo, J., & Gao, G. (2016). PlantTFDB 4.0: toward a central hub for transcription factors and regulatory interactions in plants. Nucleic acids research, gkw982.

Examples

```
data(zma.tfs)
```

Index