Package ‘MODifieRDev’

November 27, 2018

Type Package

Title MODifieRDev

Version 0.1.1

Author Dirk de Weerd

Maintainer Dirk de Weerd <dirkdeweerd.work@gmail.com>

Description An implementation of a set of disease module inference methods bundled in one package.

Imports foreach,
doParallel,
Rcpp,
dynamicTreeCut,
flashClust,
reticulate,
limma,
STRINGdb,
MODA,
AnnotationDbi,
plyr,
utils,
stats,
parallel

Depends R (>= 3.1.0),
igraph,
org.Hs.eg.db,
WGCNA

Suggests knitr,
rmarkdown

License GNU Lesser General Public License

Encoding UTF-8

LazyData true

RoxygenNote 6.1.0

LinkingTo Rcpp

SystemRequirements Python > 3, Anaconda

VignetteBuilder knitr

R topics documented:

---

add_diamond_seed_genes *Add seed genes from a DIAMOnD MODifieR_module*

Description

Add seed genes from a DIAMOnD MODifieR_module

Usage

```
add_diamond_seed_genes(diamond_module)
```

Arguments

diamond_module	A MODifieR_input object created by diamond

Details

Adds seed genes from a DIAMOnD module

Value

An object of class "MODifieR_module" with subclass "DIAMOnD"

See Also

`diamond`

`clique_sum` *Clique Sum*

Description

An implementation of the clique-based disease module inference method proposed by Barrenäs et al.

Usage

```
clique_sum(MODifieR_input, ppi_network, simplify_graph = T,
  n_iterations = 10000, clique_significance = 0.05,
  deg_cutoff = 0.05, min_clique_size = 5, min_deg_in_clique = 3,
  dataset_name = NULL)
```

Arguments

`MODifieR_input` A MODifieR input object produced by `create_input` function

`ppi_network` A PPi network as a 2 column dataframe

`simplify_graph` Remove loops and multiple edges from PPi network?

`n_iterations` Number of iterations to be performed for the permutation based p-value

`clique_significance` p-value for cliques to be considered significant

`deg_cutoff` p-value cutoff for differentialy expressed genes

`min_clique_size` Minimal size for cliques

`min_deg_in_clique` Minimum number of DEGs to be present in the clique

`dataset_name` Optional name for the input object that will be stored in the settings object. Default is the variable name of the input object

Details

The clique sum algorithm obtains *maximal cliques*, cliques that are not contained within another clique, from the `ppi_network`. The minimal size of the obtained cliques is equal to `min_clique_size`. The union of maximal cliques with a permutation based p-value below `clique_significance` and at least `min_deg_in_clique` is the disease module

Value

clique_sum returns an object of class "MODifieR_module" with subclass "Clique_Sum". This object is a named list containing the following components:

`module_genes`	A character vector containing the genes in the final module
`settings`	A named list containing the parameters used in generating the object

References

Barrenäs F, Chavali S, Alves AC, et al. Highly interconnected genes in disease-specific networks are enriched for disease-associated polymorphisms. Genome Biology. 2012;13(6):R46. doi:10.1186/gb-2012-13-6-r46.

`correlation_adjust_cutoff` *correlation_adjust_cutoff*

Description

correlation_adjust_cutoff

Usage

```
correlation_adjust_cutoff(frequency_cutoff, correlation_module)
```

Arguments

`frequency_cutoff`
: Fraction of the number of times a gene should be present in it iterations. Default is 0.5, meaning 50 procent of all iterations

`correlation_module`
: Module object that has been produced by `correlation_clique` function

Details

This function allows to adjust the frequency cutoff for a `correlation_clique` module object

Value

`correlation_clique` module object

See Also

`correlation_clique`

`correlation_clique` *Clique_correlation*

Description

A clique based method to find a disease module from correlated gene expression

Usage

```
correlation_clique(MODifieR_input, ppi_network, frequency_cutoff = 0.5,
  probabilityScaleFactor = 0.6, iteration = 50, signif_cutoff = 0.01,
  deg_cutoff = 0.05, dataset_name = NULL)
```

Arguments

`MODifieR_input`	A MODifieR input object produced by `create_input` function
`ppi_network`	A PPi network as a 2 column dataframe
`frequency_cutoff`	Fraction of the number of times a gene should be present in it iterations. Default is 0.5, meaning 50 procent of all iterations
`probabilityScaleFactor`	Scale for enriched cliques
`iteration`	Number of iterations to be performed
`signif_cutoff`	Cutoff for Fisher exact test for cliques
`deg_cutoff`	p-value cutoff for differentialy expressed genes
`dataset_name`	Optional name for the input object that will be stored in the settings object. Default is the variable name of the input object

Details

The correlation clique is a clique-based algorithm using consensus clustering. The algorithm starts with calculating a correlation score between each interaction in the PPi network. The correlation score is obtained by subtracting the Pearson correlation p-value:

$$correlationscore = 1 - Pearsonp - value$$

Subsequently, the correlation score is multiplied by the correlation confidence and scaled with the scale factor to get the edge score:

$$edgescore = \sqrt(correlationscore * confidencescore) * scale_factor$$

When the edge scores are calculated the iterative part of the algorithm commences: All edge scores are compared to random variables from the uniform distribution between (0,1) only interactions where the edge score is higher than the random variable are used to construct a new PPi network. Then, maximal cliques are inferred from this new network. The cliques are tested for significant enrichment of DEGs by Fisher's exact test and the union of significant cliques is the disease module for this iteration The final disease module will consist of genes that have been present in at least `frequency_cutoff` iterations

Value

correlation_clique returns an object of class "MODifieR_module" with subclass "Correlation_clique". This object is a named list containing the following components:

`module_genes` A character vector containing the genes in the final module

`frequency_table` A table containing the fraction of times the genes were present in an iteration module

`settings` A named list containing the parameters used in generating the object

---

`correlation_set_module_size` *correlation_set_module_size*

Description

Returns a correlation_clique module closest to `size`

Usage

```
correlation_set_module_size(size, correlation_module)
```

Arguments

`size` Module object that has been produced by `correlation_clique` function

`correlation_module` Module object that has been produced by `correlation_clique` function

Details

The function will find the the frequency cutoff for that will result in a `correlation_clique` module object closest to `size`

Value

`correlation_clique` module object

See Also

`correlation_clique`

create_custom_input_object

Create a generic input object

Description

Create a generic input object

Usage

```
create_custom_input_object(diff_genes = NULL, limma_probe_table = NULL,
  annotated_exprs_matrix = NULL, expression_matrix = NULL,
  annotation_table = NULL, group1_indici = NULL,
  group2_indici = NULL, group1_label = NULL, group2_label = NULL)
```

Arguments

`diff_genes` A 2 two column data.frame where the first column are genes and the second column are p-values

`limma_probe_table` A data.frame from `limma topTable` with added gene annotation

`annotated_exprs_matrix` A matrix where the rows are genes and the columns samples.

`expression_matrix` Normalized expression matrix where the samples are columns and probes are rows

`annotation_table` A dataframe providing annotation for the probes. The dataframe should have 3 columns:

- PROBEID: The probe id as it is in the expression matrix
- IDENTIFIER: The entrez id (if available) associated with the probe

`group1_indici` vector containing indici for samples belonging to group 1 (Column numbers)

`group2_indici` vector containing indici for samples belonging to group 2 (Column numbers)

`group1_label` Label for each group 1, for example "patient" or "control"

`group2_label` Label for each group 2, for example "patient" or "control"

Details

This function allows the creation of a generic input object with the same class as objects created by `create_input`. This can be useful in the cases where you already have differentially expressed genes, an annotated expression matrix or both and want to wrap that into an input object to use in downstream analysis.

Value

The function returns an object of class "MODifieR_input". The object is a named list containing the following components:

`diff_genes` A 2 two column data.frame where the first column are genes and the second column unadjusted p-values

`limma_probe_table`
A data.frame from `limma topTable` with added gene annotation

`annotated_exprs_matrix`
A matrix where the rows are genes and the columns samples. Probes have been collapsed into genes using `collapse_method`

`expression_matrix`
A matrix, the original input expression matrix

`annotation_table`
A data.frame, the original annotation table used to annotate the probes

`group_indici` A named list containing 2 numeric vectors. The names are the group labels and the values are the group indici

See Also

`create_input`

`create_input` *Creates an input object for downstream analysis*

Description

The MODifieR input object can be used in downstream analysis for the disease module inference methods included in this package.

Usage

```
create_input(expression_matrix, annotation_table, group1_indici,
  group2_indici, group1_label, group2_label, expression = T,
  differential_expression = T, method = "MaxMean")
```

Arguments

`expression_matrix`
Normalized expression matrix where the samples are columns and probes are rows

`annotation_table`
A dataframe providing annotation for the probes. The dataframe should have 3 columns:

- PROBEID: The probe id as it is in the expression matrix
- IDENTIFIER: The entrez id (if available) associated with the probe

`group1_indici` vector containing indici for samples belonging to group 1 (Column numbers)

`group2_indici` vector containing indici for samples belonging to group 2 (Column numbers)

`group1_label` Label for each group 1, for example "patient" or "control"

`group2_label` Label for each group 2, for example "patient" or "control"

`expression` boolean, calculate expression values?

`differential_expression`
boolean, calculate differentially expressed data?

`method` character string for determining which method is used to choose a probe among exactly 2 corresponding rows or when connectivityBasedCollapsing=FALSE. These are the options: "MaxMean" (default) or "MinMean" = choose the row with the highest or lowest mean value, respectively. "maxRowVariance" = choose the row with the highest variance (across the columns of `datET`). "absMaxMean" or "absMinMean" = choose the row with the highest or lowest mean absolute value. "ME" = choose the eigenrow (first principal component of the rows in each group). Note that with this method option, connectivityBasedCollapsing is automatically set to FALSE. "Average" = for each column, take the average value of the rows in each group "function" = use this method for a user-input function (see the description of the argument "methodFunction"). Note: if method="ME", "Average" or "function", the output parameters "group2row" and "selectedRow" are not informative.

Details

The function creates an input object to be used in all disease module inference methods. Differentially expressed genes are calculated using linear models from the `limma` package. Probes are collapsed into genes using collapseRows from `WGCNA`

Value

The function returns an object of class "MODifieR_input". The object is a named list containing the following components:

`diff_genes` A 2 two column data.frame where the first column are genes and the second column unadjusted p-values obtained by differential expression analysis

`limma_probe_table`
A data.frame from `limma topTable` with added gene annotation

`annotated_exprs_matrix`
A matrix where the rows are genes and the columns samples. Probes have been collapsed into genes using `collapse_method`

`expression_matrix`
A matrix, the original input expression matrix

`annotation_table`
A data.frame, the original annotation table used to annotate the probes

`group_indici` A named list containing 2 numeric vectors. The names are the group labels and the values are the group indici

See Also

collapseRows lmFit eBayes topTable

create_module_set *Consensus module*

Description

Consensus module

Usage

```
create_module_set(min_frequency, module_list)
```

Arguments

min_frequency	Minimal number of MODifieR modules that a gene should be present in in order to include it in the final module
module_list	A list of MODifieR modules

Details

Get a consensus module that is composed of genes present in at least min_frequency genes. If the input module_list is unnamed, the subclasses of the MODifieR objects will be used. If there is more than 1 of a given subclass present in module_list, a number will be appended.

Value

Returns an object of class "MODifieR_module" with subclass "Module_set". This object is a named list containing the following components:

module_genes	A character vector containing the genes in the final module
module_gene_list	A named list containing the module genes from the original modules
gene_frequency	Table containing all the genes present in the modules and their frequency
method_by_gene	A named list where the elements are the modules the genes have been found in and the names are the gene names
gene_by_method	A table containing the gene frequencies by combination of methods
settings	A named list containing the parameters used in generating the object

diamond *DIAMOnD module*

Description

A seed gene based algorithm to identify disease module from differentially expressed genes

Usage

```
diamond(MODifieR_input, ppi_network, deg_cutoff = 0.05,
  n_output_genes = 200, seed_weight = 10, include_seed = FALSE,
  dataset_name = NULL)
```

Arguments

`MODifieR_input`	A MODifieR input object produced by `create_input` function
`ppi_network`	A PPi network as a 2 column dataframe
`deg_cutoff`	p-value cutoff for differentialy expressed genes
`n_output_genes`	maximum number of genes to be included in the final module
`seed_weight`	Numeric additional parameter to assign weight for the seed genes
`include_seed`	Logical TRUE/FALSE for inclusion of seed genes in the output module
`dataset_name`	Optional name for the input object that will be stored in the settings object. Default is the variable name of the input object

Details

A slightly modified version of the original DIAMOnD python script is called from within R. The only change to the orginal algorithm is the option to include the seed genes to the module. There are also function to add or remove the seed genes from the output object, namely: `add_diamond_seed_genes` and `remove_diamond_seed_genes` For a detailed description of how the algorithm works, please see the paper referenced below.

Value

diamond returns an object of class "MODifieR_module" with subclass "DIAMOnD". This object is a named list containing the following components:

`module_genes`	A character vector containing the genes in the final module
`seed_genes`	Character vector containing genes that have been used as seed genes in the algorithm
`ignored_genes`	Potential seed genes that are not in the PPi network
`added_genes`	A table containing information on all added genes. First column is the name of the gene, the second column is the degree of the node (gene). The third column is the number of +1 neighbors and the fourth column is the p-value.
`settings`	A named list containing the parameters used in generating the object

References

Ghiassian, S. D., Menche, J., & Barabási, A. L. (2015). A DIseAse MOdule Detection (DIAMOnD) Algorithm Derived from a Systematic Analysis of Connectivity Patterns of Disease Proteins in the Human Interactome. PLoS Computational Biology, 11(4), 1–21. https://doi.org/10.1371/journal.pcbi.1004120

---

diffcoex *DiffCoEx*

Description

An implementation of the DiffCoEx co-expression based algorithm

Usage

```
diffcoex(MODifieR_input, beta = 6, cor_method = "spearman",
  cluster_method = "average", cuttree_method = "hybrid",
  cut_height = 0.996, deepSplit = 0, pamRespectsDendro = F,
  minClusterSize = 20, cutHeight = 0.2, dataset_name = NULL)
```

Arguments

MODifieR_input	A MODifieR input object produced by create_input function
beta	User-defined soft thresholding power For method=="tree" it defaults to 0.99. For method=="hybrid" it defaults to 99 maximum of the joining heights on the dendrogram.
cor_method	a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman": can be abbreviated.
cluster_method	the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". This applies to hierachical clustering.
cuttree_method	Chooses the method to use. Recognized values are "hybrid" and "tree".
cut_height	Maximum joining heights that will be considered.
deepSplit	For method "hybrid", can be either logical or integer in the range 0 to 4. For method "tree", must be logical. In both cases, provides a rough control over sensitivity to cluster splitting. The higher the value (or if TRUE), the more and smaller clusters will be produced. For the "hybrid" method, a finer control can be achieved via maxCoreScatter and minGap below.
pamRespectsDendro	Logical, only used for method "hybrid". If TRUE, the PAM stage will respect the dendrogram in the sense that objects and small clusters will only be assigned to clusters that belong to the same branch that the objects or small clusters being assigned belong to.

`minClusterSize`	Minimum cluster size.
`cutHeight`	Maximum joining heights that will be considered. For `method=="tree"` it defaults to 0.99. For `method=="hybrid"` it defaults to 99% of the range between the 5th percentile and the maximum of the joining heights on the dendrogram.
`dataset_name`	Optional name for the input object that will be stored in the settings object. Default is the variable name of the input object

Details

DiffCoEx is a method for identifying correlation pattern changes, which builds on the commonly used Weighted Gene Coexpression Network Analysis (WGCNA) framework for coexpression analysis.

Value

diffcoex returns an object of class "MODifieR_module" with subclass "DiffCoEx". This object is a named list containing the following components:

`module_genes`	A character vector containing the genes in the final module
`module_colors`	A character vector containing the colors that make up the final disease module
`color_vector`	A named character vector containing the genes as values and the color as name
`settings`	A named list containing the parameters used in generating the object

References

Tesson, B. M., Breitling, R., & Jansen, R. C. (2010). DiffCoEx: a simple and sensitive method to find differentially coexpressed gene modules. BMC Bioinformatics, 11, 497. `https://doi.org/10.1186/1471-2105-11-497`

`diffcoex_split_module_by_color` *Returns new DiffCoEx module objects by color*

Description

Returns new DiffCoEx module objects by color

Usage

```
diffcoex_split_module_by_color(diffcoex_module)
```

Arguments

`diffcoex_module`
: Module object that has been produced by `diffcoex` function

Details

The DiffCoEx module object is split into a series of DiffCoEx objects by color. Eevery significant color in the module will be its own DiffCoEx module object

Value

A list of DiffCoEx module objects

See Also

diffcoex

---

entrez_to_symbol *Convert the module genes in a MODifieR_input object from ENTREZ gene IDs to official gene symbols*

Description

Convert the module genes in a MODifieR_input object from ENTREZ gene IDs to official gene symbols

Usage

```
entrez_to_symbol(MODifieR_module)
```

Arguments

MODifieR_module
An object of class MODifieR_module

Details

The function uses the org.Hs.egSYMBOL function from the package org.Hs.eg.db to convert official gene symbols to ENTREZ IDs

See Also

org.Hs.egSYMBOL

symbol_to_entrez

`expression_matrix` *Expression matrix*

Description

An example dataset from GEO (accession number GSE4588) containing systemic lupus erythematosus patients and healthy controls

Usage

```
expression_matrix
```

Format

An object of class `matrix` with 52307 rows and 16 columns.

Details

The expression matrix contains 16 microarray samples. Columns 1:9 are healthy controls, columns 10:16 are SLE patients. There are 52307 probes in this dataset.

References

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE4588.

See Also

`probe_annotation`

`ppi_network`

`get_max_frequency` *Get maximal number of MODifieR modules that share at least one gene*

Description

Get maximal number of MODifieR modules that share at least one gene

Usage

```
get_max_frequency(module_list)
```

Arguments

`module_list`	A list of MODifieR modules

Value

The function returns an integer that gives the maximal number of MODifieR objects in `module_list` that share at least one gene.

`moda` *MODA*

Description

An implementation of MODA co-expression based algorithm.

Usage

```
moda(MODifieR_input, cutmethod = "Density", specificTheta = 0.1,
  conservedTheta = 0.1, dataset_name = NULL, group_of_interest)
```

Arguments

`MODifieR_input` A MODifieR input object produced by `create_input` function

`cutmethod` cutting the dendrogram based on maximal average Density or Modularity

`specificTheta` the threshold to define min(s)+specificTheta, less than which is considered as condition specific module. s is the sums of rows in Jaccard index matrix. See supplementary file.

`conservedTheta` The threshold to define max(s)-conservedTheta, greater than which is considered as condition conserved module. s is the sums of rows in Jaccard index matrix. See supplementary file.

`dataset_name` Optional name for the input object that will be stored in the settings object. Default is the variable name of the input object

`group_of_interest` Numerical value denoting which group contains the condition of interest (1 or 2)

Details

This implementation follows a workflow as described in the MODA vignette. First, two separate networks are constructed, a background network containing expression data from all samples and a condition specific network consisting of all samples minus the condition specific samples. Then, hierarchical clustering is performed and cutting height estimated from either maximal average `density` or `modularity`

Condition specific co-expression modules are then extracted using the Jaccard index and `specificTheta`.

The final module will consist of the co-expression module that has the minimal Jaccard index complemented by co-expression modules that have a Jaccard index below this minimal + `specificTheta`

After analysis, the `specificTheta` and thereby the disease module can be adjusted using `moda_change_specific_threshol`

Value

moda returns an object of class "MODifieR_module" with subclass "MODA". This object is a named list containing the following components:

`module_genes`	A character vector containing the genes in the final module
`group1_modules`	A list containing all co-expression modules in the background network
`group2_modules`	A list containing all co-expression modules in the condition specific network
`jaccard_table`	A matrix with all Jaccard indexes for all co-expression modules
`settings`	A named list containing the parameters used in generating the object

References

Li D, Brown JB, Orsini L, Pan Z, Hu G, He S (2016). MODA: MODA: MOdule Differential Analysis for weighted gene co-expression network. R package version 1.6.0

See Also

`https://bioconductor.org/packages/release/bioc/vignettes/MODA/inst/doc/MODA.html`

`moda_change_specific_threshold` *Change*

Description

Change

Usage

```
moda_change_specific_threshold(moda_module, specificTheta)
```

Arguments

`moda_module`	A `MODifieR_input` object created by `moda`
`specificTheta`	the threshold to define min(s)+specificTheta, less than which is considered as condition specific module. s is the sums of rows in Jaccard index matrix. See supplementary file.

modulediscoverer *Module Discoverer*

Description

A clique based algorithm by Vlaic et al. to produce disease module from Differentially Expressed Genes

Usage

```
modulediscoverer(MODifieR_input, ppi_network, permutations = 10000,
  deg_cutoff = 0.05, repeats = 15, clique_cutoff = 0.01,
  dataset_name = NULL)
```

Arguments

`MODifieR_input`	A MODifieR input object produced by `create_input` function
`ppi_network`	A PPi network as a 2 column dataframe
`permutations`	Number of permutations to perform to identify the community structure
`deg_cutoff`	p-value cutoff for differentialy expressed genes
`repeats`	Number of times the algorithm is repeated
`clique_cutoff`	cutoff pvalue for significant cliques
`dataset_name`	Optional name for the input object that will be stored in the settings object. Default is the variable name of the input object

Details

This is an implementation of the *single seed* Module Discoverer algorithm. The code has been adapted from the orignal code by Vlaic et al. For details, please see the paper referenced below

Value

modulediscoverer returns an object of class "MODifieR_module" with subclass "module_discoverer". This object is a named list containing the following components:

`module_genes`	A character vector containing the genes in the final module
`graph`	`igraph` graph containing the disease module
`settings`	A named list containing the parameters used in generating the object

References

Vlaic, S., Tokarski-schnelle, C., Gustafsson, M., Dahmen, U., Guthke, R., & Schuster, S. (2017). ModuleDiscoverer: Identification of regulatory modules in protein-protein interaction networks., 1–17.

See Also

https://www.leibniz-hki.de/en/modulediscoverer.html

`mod_mcode` *MCODE*

Description

A clique based algorithm to identify disease modules from differentially expressed genes originally by Bader et al

Usage

```
mod_mcode(MODifieR_input, ppi_network, hierarchy = 1, vwp = 0.5,
  haircut = F, fluff = F, fdt = 0.8, loops = T,
  deg_cutoff = 0.05, module_cutoff = 3.5, dataset_name = NULL)
```

Arguments

`MODifieR_input`	A MODifieR input object produced by `create_input` function
`ppi_network`	A PPi network as a 2 column dataframe
`hierarchy`	This parameter indicates how many hierarchy are included in the network, currently it can be `0`, 1 or 2. Default value is 1.
`vwp`	Vertex weight percentage. Default value is 0.5.
`haircut`	Boolean value, whether to remove singly-connected nodes from clusters (TRUE) or not (FALSE).
`fluff`	Boolean value, whether to spand cluster cores by one neighbour shell outwards (TRUE) or not (FALSE).
`fdt`	Cluster density cutoff. Default value is 0.8.
`loops`	Boolean value, whether to include self-loops (TRUE) or not (FALSE).
`deg_cutoff`	p-value cutoff for differentialy expressed genes
`module_cutoff`	Minimal score for a module to be returned
`dataset_name`	Optional name for the input object that will be stored in the settings object. Default is the variable name of the input object

Details

Much of the code an documentation has been taken from the now defunct package "ProNet"

Value

mcode returns a list of objects of class "MODifieR_module" with subclass "Mcode". The objects are named lists containing the following components:

`module_genes`	A character vector containing the genes in the final module
`module_scores`	A numeric value that denotes the score of the module. Higher is better
`settings`	A named list containing the parameters used in generating the object

References

Bader, G. D., & Hogue, C. W. (2003). An automated method for finding molecular complexes in large protein interaction networks. BMC Bioinformatics, 4(1), 2. https://doi.org/10.1186/1471-2105-4-2

See Also

https://github.com/cran/ProNet

---

ppi_network *PPi network*

Description

A Protein-Protein interaction network from STRING

Usage

```
ppi_network
```

Format

An object of class `data.frame` with 64672 rows and 3 columns.

Details

This PPi network from STRING is version 7.1 and filtered for interactions with a confidence score higher than 700. The identifiers for the genes are ENTREZ. There are 64672 interactions (rows) in the dataframe. Each row denotes an interaction; column 1 contains the first gene, column 2 the second. The third column is the confidence score.

References

https://string-db.org/

See Also

`probe_annotation`

`expression_matrix`

`probe_annotation` *Probe annotation*

Description

Probe annotation for the example dataset `expression_matrix`. The annotation is taken from Bioconductor package hgu133plus2.db and contains 52307 probes.

Usage

```
probe_annotation
```

Format

An object of class `data.frame` with 52307 rows and 2 columns.

References

`https://bioconductor.org/packages/release/data/annotation/html/hgu133plus2.db.html`

Carlson M (2016). hgu133plus2.db: Affymetrix Human Genome U133 Plus 2.0 Array annotation data (chip hgu133plus2). R package version 3.2.3.

See Also

`expression_matrix`

`ppi_network`

`remove_diamond_seed_genes` *Remove seed genes from a DIAMOnD MODifieR_module*

Description

Remove seed genes from a DIAMOnD MODifieR_module

Usage

```
remove_diamond_seed_genes(diamond_module)
```

Arguments

`diamond_module`	A `MODifieR_input` object created by `diamond`

Details

Removes seed genes from a DIAMOnD module

Value

An object of class "MODifieR_module" with subclass "DIAMOnD"

See Also

diamond

---

symbol_to_entrez *Convert the module genes in a MODifieR_input object from official gene symbols to ENTREZ gene IDs*

Description

Convert the module genes in a MODifieR_input object from official gene symbols to ENTREZ gene IDs

Usage

```
symbol_to_entrez(MODifieR_module)
```

Arguments

`MODifieR_module`
An object of class MODifieR_module

Details

The function uses the `org.Hs.egSYMBOL` function from the package `org.Hs.eg.db` to convert ENTREZ IDs to official gene symbols

See Also

org.Hs.egSYMBOL

entrez_to_symbol

wgcna *An implementation of WGCNA to correlate coexpression modules to disease*

Description

An implementation of WGCNA to correlate coexpression modules to disease

Usage

```
wgcna(MODifieR_input, minModuleSize = 30, deepSplit = 2,
  pamRespectsDendro = F, mergeCutHeight = 0.1, numericLabels = T,
  pval_cutoff = 0.05, saveTOMs = T,
  dataset_name = deparse(substitute(MODifieR_input)))
```

Arguments

`MODifieR_input` A MODifieR input object produced by `create_input` function

`minModuleSize` minimum module size for module detection. See `cutreeDynamic` for more details.

`deepSplit` integer value between 0 and 4. Provides a simplified control over how sensitive module detection should be to module splitting, with 0 least and 4 most sensitive. See `cutreeDynamic` for more details.

`pamRespectsDendro` Logical, only used when `pamStage` is `TRUE`. If `TRUE`, the PAM stage will respect the dendrogram in the sense an object can be PAM-assigned only to clusters that lie below it on the branch that the object is merged into. See `cutreeDynamic` for more details.

`mergeCutHeight` dendrogram cut height for module merging.

`numericLabels` logical: should the returned modules be labeled by colors (`FALSE`), or by numbers (`TRUE`)?

`pval_cutoff` The p-value cutoff to be used for significant co-expression modules (colors)

`saveTOMs` logical: should the consensus topological overlap matrices for each block be saved and returned?

`dataset_name` Optional name for the input object that will be stored in the settings object. Default is the variable name of the input object

Details

wgcna is an implementation of WGCNA that associates co-expression modules (denoted by color) to a trait. Co-expression modules with an adjusted p-value < `pval_cutoff` will make up the final disease module.

The algorithm infers co-expression modules from combined expression dataset from both `group1` and `group2`. Co-expression modules are then correlated to trait (group 1 ~ group 2).

After analysis there are some post-processing functions available:

- wgcna_get_all_module_genes Get a list with all genes sorted by module color
- wgcna_get_module_genes_by_sign Get a module with either only postively correlated genes or negatively correlated genes
- wgcna_adjust_significance Adjust p-value cutoff
- wgcna_split_module_by_color Get a list where each color is a separate module
- wgcna_set_module_size Get a module close to a specific size

Value

wgcna returns an object of class "MODifieR_module" with subclass "WGCNA". This object is a named list containing the following components:

module_genes	A character vector containing the genes in the final module
info_table	A data.frame containing all genes and their assigned colors
correlation_to_trait_table	A data.frame containing all module colors and their p- and adjusted p-value
softthreshold_value	A numeric, the soft threshold power that is used. See: pickSoftThreshold
module_colors	A character vector containing the colors that make up the final disease module
settings	A named list containing the parameters used in generating the object

References

Langfelder P and Horvath S, WGCNA: an R package for weighted correlation network analysis. BMC Bioinformatics 2008, 9:559 doi:10.1186/1471-2105-9-559

Peter Langfelder, Steve Horvath (2012). Fast R Functions for Robust Correlations and Hierarchical Clustering. J ournal of Statistical Software, 46(11), 1-17. URL http://www.jstatsoft.org/v46/i11/

wgcna_adjust_significance

wgcna_adjust_significance

Description

wgcna_adjust_significance

Usage

```
wgcna_adjust_significance(pval_cutoff, wgcna_module, use_unadjusted = F)
```

Arguments

pval_cutoff	The p-value cutoff to be used for significant co-expression modules (colors)
wgcna_module	Module object that has been produced by wgcna function
use_unadjusted	Boolean value to signify if the adjusted (TRUE) or unadjusted (FALSE) p value should be used to adjust significance

Details

This function allows to adjust the significance cutoff for a `wgcna` module object

Value

`wgcna` module object

See Also

`wgcna`

`wgcna_get_all_module_genes`
Generate a list of module colors with their respective genes.

Description

Generate a list of module colors with their respective genes.

Usage

```
wgcna_get_all_module_genes(wgcna_module)
```

Arguments

`wgcna_module` Module object that has been produced by `wgcna` function

Value

Returns a `named list` of module genes where the names are the module colors. Includes non-significant colors

See Also

`wgcna`

wgcna_get_module_genes_by_sign *Split WGCNA module in module containing only positive or negative correlation*

Description

Split WGCNA module in module containing only positive or negative correlation

Usage

```
wgcna_get_module_genes_by_sign(wgcna_module, mode)
```

Arguments

wgcna_module	Module object that has been produced by wgcna function
mode	Character. "p" or "positive" for positive correlation, "n" or "negative" for negative correlation.

Details

The functions returns a new wgcna module object that only contains positively or negatively correlated colors

Value

wgcna_module object

See Also

wgcna

wgcna_set_module_size *Returns a wgcna module closest to* size

Description

Returns a wgcna module closest to size

Usage

```
wgcna_set_module_size(size, wgcna_module)
```

Arguments

size	The desired size of the resulting module
wgcna_module	Module object that has been produced by wgcna function

Details

The function starts with the co-expression module (color) with the lowest p-value and gradually adds more co-expression modules until the module will be as close as possible to size `size`

@return

`wgcna` module object

See Also

`wgcna`

`wgcna_split_module_by_color`

Returns new WGCNA module objects by color

Description

Returns new WGCNA module objects by color

Usage

```
wgcna_split_module_by_color(wgcna_module)
```

Arguments

`wgcna_module`	Module object that has been produced by `wgcna` function

Details

The `wgcna` module object is split into a series of `wgcna` objects by color. Eevery significant color in the module will be its own `wgcna` module object

Value

A list of `wgcna` module objects

See Also

`wgcna`

Index