Package 'multiDEGGs'

January 16, 2026

Title Multi-Omic Differentially Expressed Gene-Gene Pairs

Version 1.1.2

Maintainer Elisabetta Sciacca <e.sciacca@qmul.ac.uk>

Description Performs multi-omic differential network analysis by revealing differential interactions between molecular entities (genes, proteins, transcription factors, or other biomolecules) across the omic datasets provided.
For each omic dataset, a differential network is constructed where links represent statistically significant differential interactions between entities. These networks are then integrated into a comprehensive visualization using distinct colors to distinguish interactions from different omic layers. This unified display allows interactive exploration of cross-omic patterns, such as differential interactions present at both transcript and protein levels. For each link, users can access differential statistical significance metrics (p values or adjusted p values, calculated via robust or traditional linear regression with interaction term) and differential regression plots.
The methods implemented in this package are described in Sciacca et al. (2023) <doi:10.1093/bioinformatics/btad192>.

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression gzip

RoxygenNote 7.3.3

Language en-gb

URL https://github.com/elisabettasciacca/multiDEGGs/, https://elisabettasciacca.github.io/multiDEGGs/

BugReports https://github.com/elisabettasciacca/multiDEGGs/issues

Suggests kernlab, nestedcv, pls, qvalue, randomForest, ranger, testthat (>= 3.0.0)

Imports DT, grDevices, graphics, knitr, MASS, magrittr, methods, parallel, pbapply, pbmcapply, rmarkdown, sfsmisc, shiny, shinydashboard, stats, utils, visNetwork

Depends R (>= 4.4.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Elisabetta Sciacca [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-7525-1558>), Myles Lewis [ctb] (ORCID: <https://orcid.org/0000-0001-9365-5345>)

Repository CRAN

Date/Publication 2026-01-16 00:40:02 UTC

Contents

`.predict_multiDEGGs` *Predict method for multiDEGGs_filter objects*

Description

This function generates predictions by creating a dataset with single and combined predictors based on the filtering results of a multiDEGGs_filter model.

Usage

```
.predict_multiDEGGs(
  object,
  newdata,
  interaction.type = "ratio",
  sep = ":",
  ...
)
```

Arguments

`object`	A fitted object of class `multiDEGGs_filter` containing filtering results with: **keep** Character vector of variable names to keep as single predictors **pairs** Data frame or matrix with two columns specifying pairs of variables to combine
`newdata`	A data frame containing the new data for prediction. Must contain all variables specified in `object$keep` and `object$pairs`.
`interaction.type`	Character string specifying how to combine the paired predictors. Options are: **"ratio"** Combine paired predictors by dividing the first variable by the second (a/b) **other** Combine paired predictors by multiplying the variables (a*b) Default is "ratio".
`sep`	Character string used as separator when creating column names for combined predictors. Default is ":".
`...`	Additional arguments passed to the generic function.

Details

The function processes the filtering results in two steps:

1. Selects single predictors from `newdata` based on variables listed in `object$keep`
2. Adds combined predictors from paired variables in `object$pairs`

Value

A data frame containing:

- Single predictors (if any are specified in `object$keep`)
- Combined predictors based on variable pairs and interaction type

---

`calc_pvalues_network` *Calculate the p values for specific category network samples*

Description

Calculate the p values for specific category network samples

Usage

```
calc_pvalues_network(
  assayData,
  metadata,
  padj_method,
  categories_length,
  regression_method = "lm",
  category_network
)
```

Arguments

`assayData`	a matrix or data.frame (or list of matrices or data.frames for multi-omic analysis) containing normalised assay data. Sample IDs must be in columns and probe IDs (genes, proteins...) in rows. For multi omic analysis, it is highly recommended to use a named list of data. If unnamed, sequential names (assayData1, assayData2, etc.) will be assigned to identify each matrix or data.frame.
`metadata`	a named vector, matrix, or data.frame containing sample annotations or categories. If matrix or data.frame, each row should correspond to a sample, with columns representing different sample characteristics (e.g., treatment group, condition, time point). The colname of the sample characteristic to be used for differential analysis must be specified in `category_variable`. Rownames must match the sample IDs used in assayData. If named vector, each element must correspond to a sample characteristic to be used for differential analysis, and names must match sample IDs used in the colnames of `assayData`. Continuous variables are not allowed.
`padj_method`	a character string indicating the p values correction method for multiple test adjustment. It can be either one of the methods provided by the `p.adjust` function from `stats` (bonferroni, BH, hochberg, etc.) or "q.value" for Storey's q values, or "none" for unadjusted p values. When using "q.value" the `qvalue` package must be installed first.

`categories_length`
: integer number indicating the number of categories

`regression_method`
: whether to use robust linear modelling to calculate link p values. Options are 'lm' (default) or 'rlm'. The lm implementation is faster and lighter.

`category_network`
: network table for a specific category

Value

a list of p values

`calc_pvalues_percentile` *Compute interaction p values for a single percentile value*

Description

Compute interaction p values for a single percentile value

Usage

```
calc_pvalues_percentile(
  assayData,
  metadata,
  categories_length,
  category_median_list,
  padj_method,
  percentile,
  contrasts,
  regression_method,
  edges,
  sig_edges_count
)
```

Arguments

`assayData`
: a matrix or data.frame (or list of matrices or data.frames for multi-omic analysis) containing normalised assay data. Sample IDs must be in columns and probe IDs (genes, proteins...) in rows. For multi omic analysis, it is highly recommended to use a named list of data. If unnamed, sequential names (assayData1, assayData2, etc.) will be assigned to identify each matrix or data.frame.

`metadata`
: a named vector, matrix, or data.frame containing sample annotations or categories. If matrix or data.frame, each row should correspond to a sample, with columns representing different sample characteristics (e.g., treatment group, condition, time point). The colname of the sample characteristic to be used for differential analysis must be specified in `category_variable`. Rownames must

match the sample IDs used in assayData. If named vector, each element must correspond to a sample characteristic to be used for differential analysis, and names must match sample IDs used in the colnames of `assayData`. Continuous variables are not allowed.

`categories_length`
: integer number indicating the number of categories

`category_median_list`
: list of category data.frames

`padj_method`
: a character string indicating the p values correction method for multiple test adjustment. It can be either one of the methods provided by the `p.adjust` function from `stats` (bonferroni, BH, hochberg, etc.) or "q.value" for Storey's q values, or "none" for unadjusted p values. When using "q.value" the `qvalue` package must be installed first.

`percentile`
: a float number indicating the percentile to use.

`contrasts`
: data.frame containing the categories contrasts in rows

`regression_method`
: whether to use robust linear modelling to calculate link p values. Options are 'lm' (default) or 'rlm'. The lm implementation is faster and lighter.

`edges`
: network of biological interactions in the form of a table of class data.frame with two columns: "from" and "to".

`sig_edges_count`
: number of significant edges ($p < 0.05$)

Value

The list of float numbers of the significant pvalues for a single percentile

---

`cat_parallel` *cat_parallel (from nestedcv)*

Description

Prints using shell echo from inside mclapply when run in Rstudio

Usage

```
cat_parallel(...)
```

Arguments

`...`
: to be passed to system()

`get_diffNetworks` *Generate multi-omic differential networks*

Description

Generate a multi-layer differential network with interaction p values

Usage

```
get_diffNetworks(
  assayData,
  metadata,
  category_variable = NULL,
  regression_method = "lm",
  category_subset = NULL,
  network = NULL,
  percentile_vector = seq(0.35, 0.98, by = 0.05),
  padj_method = "bonferroni",
  show_progressBar = TRUE,
  verbose = TRUE,
  cores = parallel::detectCores()/2
)
```

Arguments

`assayData`	a matrix or data.frame (or list of matrices or data.frames for multi-omic analysis) containing normalised assay data. Sample IDs must be in columns and probe IDs (genes, proteins...) in rows. For multi omic analysis, it is highly recommended to use a named list of data. If unnamed, sequential names (assayData1, assayData2, etc.) will be assigned to identify each matrix or data.frame.
`metadata`	a named vector, matrix, or data.frame containing sample annotations or categories. If matrix or data.frame, each row should correspond to a sample, with columns representing different sample characteristics (e.g., treatment group, condition, time point). The colname of the sample characteristic to be used for differential analysis must be specified in `category_variable`. Rownames must match the sample IDs used in assayData. If named vector, each element must correspond to a sample characteristic to be used for differential analysis, and names must match sample IDs used in the colnames of `assayData`. Continuous variables are not allowed.
`category_variable`	when metadata is a matrix or data.frame this is the column name of `metadata` that contains the sample annotations to be used for differential analysis
`regression_method`	whether to use robust linear modelling to calculate link p values. Options are 'lm' (default) or 'rlm'. The lm implementation is faster and lighter.

`category_subset`
: optional character vector indicating a subset of categories from the category variable. If not specified, all categories in `category_variable` will be used.

`network`
: network of biological interactions provided by the user. The network must be provided in the form of a table of class data.frame with only two columns named "from" and "to". If NULL (default) a network of 10,537 molecular interactions obtained from KEGG, mirTARbase, miRecords and transmiR will be used. This has been obtained via the `exportgraph` function of the MITHrIL tool (Alaimo et al., 2016).

`percentile_vector`
: a numeric vector specifying the percentiles to be used in the percolation analysis. By default, it is defined as `seq(0.35, 0.98, by = 0.05)`, which generates a sequence of percentiles starting at 0.35, meaning that targets (genes/proteins...) whose expression value is under the 35th percentile of the whole matrix will be excluded. This threshold can be modified by specifying a different starting point for `seq`. For a more granular percolation analysis an higher optimisation of the algorithm, `by = 0.05` can be modified in favour of lower values, but this will increase the computational time.

`padj_method`
: a character string indicating the p values correction method for multiple test adjustment. It can be either one of the methods provided by the `p.adjust` function from `stats` (bonferroni, BH, hochberg, etc.) or "q.value" for Storey's q values, or "none" for unadjusted p values. When using "q.value" the `qvalue` package must be installed first.

`show_progressBar`
: logical. Whether to display a progress bar during execution. Default is TRUE.

`verbose`
: logical. Whether to print detailed output messages during processing. Default is TRUE

`cores`
: number of cores to use for parallelisation.

Value

a `deggs` object containing differential networks incorporating p values or adjusted p values for each link.

Examples

```
# Single omic analysis:
data("synthetic_metadata")
data("synthetic_rnaseqData")
deggs_object <- get_diffNetworks(assayData = synthetic_rnaseqData,
                                 metadata = synthetic_metadata,
                                 category_variable = "response",
                                 regression_method = "lm",
                                 padj_method = "bonferroni",
                                 verbose = FALSE,
                                 show_progressBar = FALSE,
                                 cores = 1)

# multi-omic analysis:
```

```
data("synthetic_metadata")
data("synthetic_rnaseqData")
data("synthetic_proteomicData")
data("synthetic_OlinkData")
assayData_list <- list("RNAseq" = synthetic_rnaseqData,
                       "Proteomics" = synthetic_proteomicData,
                       "Olink" = synthetic_OlinkData)
deggs_object <- get_diffNetworks(assayData = assayData_list,
                                 metadata = synthetic_metadata,
                                 category_variable = "response",
                                 regression_method = "lm",
                                 padj_method = "bonferroni",
                                 verbose = FALSE,
                                 show_progressBar = FALSE,
                                 cores = 1)

# to use only certain categories for comparison:
# let's randomly add another level of response to the example metadata
synthetic_metadata$response <- as.character(synthetic_metadata$response)
indices <- sample(1:nrow(synthetic_metadata), 20, replace = FALSE)
synthetic_metadata$response[indices] <- "Moderate response"
deggs_object <- get_diffNetworks(assayData = assayData_list,
                                 metadata = synthetic_metadata,
                                 category_variable = "response",
                                 category_subset = c("Responder",
                                                     "Non_responder"),
                                 regression_method = "lm",
                                 verbose = FALSE,
                                 show_progressBar = FALSE,
                                 cores = 1)

# to be more generous on the targets to be excluded, and lower the expression
# level threshold to the 25th percentile (or lower):
deggs_object <- get_diffNetworks(assayData = assayData_list,
                                 metadata = synthetic_metadata,
                                 category_variable = "response",
                                 category_subset = c("Responder",
                                                     "Non_responder"),
                                 regression_method = "lm",
                                 percentile_vector = seq(0.25, 0.98, by = 0.05),
                                 verbose = FALSE,
                                 show_progressBar = FALSE,
                                 cores = 1)
```

`get_diffNetworks_singleOmic` *Generate differential networks for single omic analysis*

Description

Generate differential networks for single omic analysis

Usage

```
get_diffNetworks_singleOmic(
  assayData,
  assayDataName,
  metadata,
  regression_method,
  network,
  percentile_vector,
  padj_method,
  show_progressBar,
  verbose,
  cores
)
```

Arguments

assayData	a matrix or data.frame (or list of matrices or data.frames for multi-omic analysis) containing normalised assay data. Sample IDs must be in columns and probe IDs (genes, proteins...) in rows. For multi omic analysis, it is highly recommended to use a named list of data. If unnamed, sequential names (assayData1, assayData2, etc.) will be assigned to identify each matrix or data.frame.
assayDataName	name of the assayData, to identify which omic is.
metadata	a named vector, matrix, or data.frame containing sample annotations or categories. If matrix or data.frame, each row should correspond to a sample, with columns representing different sample characteristics (e.g., treatment group, condition, time point). The colname of the sample characteristic to be used for differential analysis must be specified in `category_variable`. Rownames must match the sample IDs used in assayData. If named vector, each element must correspond to a sample characteristic to be used for differential analysis, and names must match sample IDs used in the colnames of `assayData`. Continuous variables are not allowed.
regression_method	whether to use robust linear modelling to calculate link p values. Options are 'lm' (default) or 'rlm'. The lm implementation is faster and lighter.
network	network of biological interactions provided by the user. The network must be provided in the form of a table of class data.frame with only two columns named "from" and "to". If NULL (default) a network of 10,537 molecular interactions obtained from KEGG, mirTARbase, miRecords and transmiR will be used. This has been obtained via the `exportgraph` function of the MITHrIL tool (Alaimo et al., 2016).
percentile_vector	a numeric vector specifying the percentiles to be used in the percolation analysis. By default, it is defined as `seq(0.35, 0.98, by = 0.05)`, which generates a sequence of percentiles starting at 0.35, meaning that targets (genes/proteins...) whose expression value is under the 35th percentile of the whole matrix will be excluded. This threshold can be modified by specifying a different starting point for `seq`. For a more granular percolation analysis an higher optimisation of the

algorithm, by = 0.05 can be modified in favour of lower values, but this will increase the computational time.

padj_method — a character string indicating the p values correction method for multiple test adjustment. It can be either one of the methods provided by the p.adjust function from stats (bonferroni, BH, hochberg, etc.) or "q.value" for Storey's q values, or "none" for unadjusted p values. When using "q.value" the qvalue package must be installed first.

show_progressBar — logical. Whether to display a progress bar during execution. Default is TRUE.

verbose — logical. Whether to print detailed output messages during processing. Default is TRUE

cores — number of cores to use for parallelisation.

Value

a list of differential networks, one per category

---

get_multiOmics_diffNetworks

Get a table of all significant interactions across categories

---

Description

Get a table of all significant interactions across categories

Usage

```
get_multiOmics_diffNetworks(deggs_object, sig_threshold = 0.05)
```

Arguments

deggs_object — an object of class deggs generated by get_diffNetworks

sig_threshold — threshold for significance. Default 0.05.

Value

a list of multilayer networks (as edge tables), one per category.

Examples

```
data("synthetic_metadata")
data("synthetic_rnaseqData")
data("synthetic_proteomicData")
data("synthetic_OlinkData")
assayData_list <- list("RNAseq" = synthetic_rnaseqData,
                       "Proteomics" = synthetic_proteomicData,
                       "Olink" = synthetic_OlinkData)
```

```
deggs_object <- get_diffNetworks(assayData = assayData_list,
                                 metadata = synthetic_metadata,
                                 category_variable = "response",
                                 verbose = FALSE,
                                 show_progressBar = FALSE,
                                 cores = 2)
get_multiOmics_diffNetworks(deggs_object, sig_threshold = 0.05)
```

`get_sig_deggs` *Get a table of all the significant interactions across categories*

Description

Get a table of all the significant interactions across categories

Usage

```
get_sig_deggs(deggs_object, assayDataName = 1, sig_threshold = 0.05)
```

Arguments

`deggs_object`	an object of class `deggs` generated by `get_diffNetworks`
`assayDataName`	name of the assayData of interest. If an unnamed list of data was given to `get_diffNetworks`, `assayDataName` here will be the number corresponding to the position of the data in the `assayDataList` provided before (i.e. if transcriptomic data was second in the list, a list of all its differential interactions can be obtained with assayDataName = 2, if only one data table was provided assayDataName must be 1). Default 1.
`sig_threshold`	threshold for significance. Default 0.05.

Value

a `data.frame` listing all the significant differential interactions found across categories for that particular omic data. This list can also be used to substitute or integrate feature selection in machine learning models for the prediction of the categories (see vignette).

Examples

```
data("synthetic_metadata")
data("synthetic_rnaseqData")
deggs_object <- get_diffNetworks(assayData = synthetic_rnaseqData,
                                 metadata = synthetic_metadata,
                                 category_variable = "response",
                                 verbose = FALSE,
                                 show_progressBar = FALSE,
                                 cores = 2)
get_sig_deggs(deggs_object, sig_threshold = 0.05)
```

multiDEGGs_combined_filter *Combined multiDEGGs filter*

Description

This function can be passed to the `modifyX` parameter of nestcv.train or nestcv.glmnet to use one of the available statistical filters (t-test, wilcoxon, etc.) in combination with multiDEGGs. Single predictors will be selected by the selected statistical filter an paired predictors will be added by multiDEGGs.

Usage

```
multiDEGGs_combined_filter(
  y,
  x,
  filter_method = "ttest",
  nfilter,
  dynamic_nfilter = TRUE,
  keep_single_genes = FALSE,
  ...
)
```

Arguments

`y`	Numeric vector or factor. Response variable (outcome), i.e. the 'metadata' named vector, as passed by nestcv.train or nestcv.glmnet.
`x`	Predictor variables, i.e. the assayData matrix with genes in columns and IDs in rows, as passed by nestcv.train or nestcv.glmnet.
`filter_method`	Character string. Statistical filtering method to be used in combination with multiDEGGs for sigle feature selection. Options are: "ttest", "wilcoxon", "ranger", "glmnet", "pls".
`nfilter`	Integer. Maximum number of features to select.
`dynamic_nfilter`	Logical. If `TRUE` `nfilter` will limit the number of features selected by the statistical filter and the feature space will be augmented by adding ALL the paired predictors found by multiDEGGs. If `FALSE` `nfilter` will limit the total number of predictors, with approximately half allocated to pairs and half to single genes.
`keep_single_genes`	Logical. When `dynamic_nfilter = TRUE`, determines whether to include single genes selected by multiDEGGs (i.e. the single variables included in the differential pairs) in addition to those from the statistical filter. Default is FALSE.
`...`	Additional arguments passed to the filtering functions.

Details

The function operates in two modes:

Dynamic Filtering (dynamic_nfilter = TRUE):

- Selects `nfilter` single genes using the specified statistical method
- Finds all significant gene pairs using multiDEGGs
- Total predictors = nfilter single genes + number of significant pairs
  - If `keep_single_genes = TRUE`, also includes single genes obtained from pairs found by multiDEGGs

Balanced Selection (dynamic_nfilter = FALSE):

- Allocates approximately half of `nfilter` to gene pairs
- Remaining slots filled with single genes from the statistical filter
- If fewer pairs are found than allocated, compensates by selecting more single genes
- Ensures consistent total number of predictors across outer folds

The statistical filtering methods include:

- `"ttest"`: Two-sample t-test for differential expression
- `"wilcoxon"`: Wilcoxon rank-sum test
- `"ranger"`: Random Forest variable importance
- `"glmnet"`: Elastic net regularization
- `"pls"`: Partial Least Squares variable importance

Value

An object of class "multiDEGGs_filter" containing:

`keep`	Character vector of selected single gene names
`pairs`	Data frame of selected gene pairs with interaction information

Examples

```
library(nestedcv)
data("synthetic_metadata")
data("synthetic_rnaseqData")

# fit a regularized linear model
# note that nfilter, n_outer_folds, n_inner_folds are set low to keep the
# example lightweight. Adjust these values as needed for your use case.
## Not run:
fit.glmnet <- nestedcv::nestcv.glmnet(
y = as.numeric(synthetic_metadata$response),
x =  t(synthetic_rnaseqData),
modifyX = "multiDEGGs_combined_filter",
modifyX_options = list(filter_method = "ttest",
                       nfilter = 5,
```

```
                        dynamic_nfilter = TRUE,
                        keep_single_genes = FALSE),
  modifyX_useY = TRUE,
  n_outer_folds = 4,
  n_inner_folds = 4)

  summary(fit.glmnet)

  ## End(Not run)

  # fit a random forest model
  # NOTE: nfilter, n_outer_folds, n_inner_folds are set low to keep the
  # example lightweight. Adjust these values as needed for your use case.
  fit.rf <- nestedcv::nestcv.train(
    y = synthetic_metadata$response,
    x = t(synthetic_rnaseqData),
    method = "rf",
    modifyX = "multiDEGGs_combined_filter",
    modifyX_options = list(filter_method = "ttest",
                           nfilter = 5,
                           dynamic_nfilter = TRUE,
                           keep_single_genes = FALSE),
    modifyX_useY = TRUE,
    n_outer_folds = 2,
    n_inner_folds = 2
  )

  fit.rf$summary
```

`multiDEGGs_filter` *multiDEGGs_filter*

Description

Function to be passed to the `modifyX` parameter of nestcv.train or nestcv.glmnet to allow nested feature selection and augmentation via differential network analysis with multiDEGGs.

Usage

```
multiDEGGs_filter(y, x, keep_single_genes = FALSE, nfilter = NULL)
```

Arguments

`y`	Numeric vector or factor. Response variable (outcome), i.e. the 'metadata' named vector, as passed by nestcv.train or nestcv.glmnet.
`x`	Predictor variables, i.e. the assayData matrix with genes in columns and IDs in rows, as passed by nestcv.train or nestcv.glmnet.

keep_single_genes
: Logical, default FALSE. If TRUE, the function will return unique individual genes along with significant pairs.

nfilter
: Integer. Maximum total number of predictors to return. When keep_single_genes = TRUE, this parameter limits the combined count of unique and paired predictors (i.e., length(keep_DEGGs) + nrow(pairs) <= nfilter). Predictors are included from most to least significant: for each pair, both the pair itself and the new unique variables are included until the nfilter threshold is reached. When keep_single_genes = FALSE, nfilter only limits the number of pairs returned. If NULL, no filtering is applied and the total number of predictors will depend on how many significantly different interactions are detected by multiDEGGs in that fold.

Value

a list containing two types of predictors:

- single predictors (stored in the 'keep_DEGGs' vector)
- paired predictors (stored in the 'pairs' dataframe) Note that nfilter limits the maximum number of engineered features returned, however this number might be lower and will depend on how many significantly different interactions will be found in each fold by multiDEGGs. **If no significantly different interactions are found the function will print a '0' and switch to t-test for that fold.**

Examples

```
library(nestedcv)
data("synthetic_metadata")
data("synthetic_rnaseqData")

# fit a regularized linear model
# Note that nfilter, n_outer_folds, n_inner_folds are set low to keep the
# example lightweight. Adjust these values as needed for your use case.
## Not run:
fit.glmnet <- nestcv.glmnet(
  y = as.numeric(synthetic_metadata$response),
  x =  t(synthetic_rnaseqData),
  modifyX = "multiDEGGs_filter",
  modifyX_options = list(keep_single_genes = FALSE,
                         nfilter = 5),
  modifyX_useY = TRUE,
  n_outer_folds = 4,
  n_inner_folds = 4)

summary(fit.glmnet)

## End(Not run)

# fit a random forest model:
# note that nfilter, n_outer_folds, n_inner_folds are set low to keep the
# example lightweight. Adjust these values as needed for your use case.
```

```
fit.rf <- nestcv.train(
  y = synthetic_metadata$response,
  x = t(synthetic_rnaseqData),
  method = "rf",
  modifyX = "multiDEGGs_filter",
  modifyX_options = list(keep_single_genes = FALSE,
                         nfilter = 5),
  modifyX_useY = TRUE,
  n_outer_folds = 2,
  n_inner_folds = 2
)

fit.rf$summary
```

---

my_palette *Internal function for colors*

Description

This function return a color palette with the number of colors specified by n

Usage

```
my_palette(n)
```

Arguments

n	number of colors needed

Value

a vector with colors

---

node_boxplot *Boxplots of single nodes (genes,proteins, etc.)*

Description

This function is for internal use of `View_diffnetworks`

Usage

```
node_boxplot(gene, assayDataName = 1, deggs_object)
```

Arguments

gene	gene name (must be in rownames(assayData))
assayDataName	name of the assayData of interest. If an unnamed list of data was given to get_diffNetworks, the assayDataName here will be the number indicating the position of the data in the assayDataList provided before (i.e. if the user wants to plot a differential interaction observed in the transcriptomic data, which was second in the list, then assayDataName must be 2, if only one data table was provided assayDataName must be 1). Default 1.
deggs_object	an object of class deggs generated by get_diffNetworks

Value

the boxplot

---

plot_regressions *Plot differential regressions for a link*

Description

Plot differential regressions for any target-target pair in an omic dataset

Usage

```
plot_regressions(
  deggs_object,
  assayDataName = 1,
  gene_A,
  gene_B,
  title = NULL,
  legend_position = "topright"
)
```

Arguments

deggs_object	an object of class deggs generated by get_diffNetworks
assayDataName	name of the assayData of interest. If an unnamed list of data was given to get_diffNetworks, the assayDataName here will be the number indicating the position of the data in the assayDataList provided before (i.e. if the user wants to plot a differential interaction observed in the transcriptomic data, which was second in the list, then assayDataName must be 2, if only one data table was provided assayDataName must be 1). Default 1.
gene_A	character. Name of the first target (gene, protein, metabolite, etc.)
gene_B	character. Name of the second target (gene, protein, metabolite, etc.)
title	plot title. If NULL (default), the name of the assayData will be used. Use empty character "" for no title.

`legend_position`
: position of the legend in the plot. It can be specified by keyword or in any parameter accepted by `xy.coords` (defalut "topright")

Value

base graphics plot showing differential regressions across categories. The p value of the interaction term of gene A ~ gene B * category is reported on top.

Examples

```
data("synthetic_metadata")
data("synthetic_rnaseqData")
data("synthetic_proteomicData")
data("synthetic_OlinkData")
assayData_list <- list("RNAseq" = synthetic_rnaseqData,
                       "Proteomics" = synthetic_proteomicData,
                       "Olink" = synthetic_OlinkData)
deggs_object <- get_diffNetworks(assayData = assayData_list,
                                 metadata = synthetic_metadata,
                                 category_variable = "response",
                                 regression_method = "lm",
                                 padj_method = "bonferroni",
                                 verbose = FALSE,
                                 show_progressBar = FALSE,
                                 cores = 1)
plot_regressions(deggs_object,
                 assayDataName = "RNAseq",
                 gene_A = "MTOR",
                 gene_B = "AKT2",
                 legend_position = "bottomright")
```

---

predict.multiDEGGs_filter

Wrapper of .predict_multiDEGGs for multiDEGGs_filter()

Description

This function generates predictions by creating a dataset with single and combined predictors based on the filtering results of a multiDEGGs_filter model.

Usage

```
## S3 method for class 'multiDEGGs_filter'
predict(object, newdata, interaction.type = "ratio", sep = ":", ...)
```

Arguments

`object`	A fitted object of class `multiDEGGs_filter` containing filtering results with: **keep** Character vector of variable names to keep as single predictors **pairs** Data frame or matrix with two columns specifying pairs of variables to combine
`newdata`	A data frame containing the new data for prediction. Must contain all variables specified in `object$keep` and `object$pairs`.
`interaction.type`	Character string specifying how to combine the paired predictors. Options are: **"ratio"** Combine paired predictors by dividing the first variable by the second (a/b) **other** Combine paired predictors by multiplying the variables (a*b) Default is "ratio".
`sep`	Character string used as separator when creating column names for combined predictors. Default is ":".
`...`	Additional arguments passed to the generic function.

Details

The function processes the filtering results in two steps:

1. Selects single predictors from `newdata` based on variables listed in `object$keep`
2. Adds combined predictors from paired variables in `object$pairs`

Value

A data frame containing:

- Single predictors (if any are specified in `object$keep`)
- Combined predictors based on variable pairs and interaction type

---

`predict.multiDEGGs_filter_combined` *Wrapper of .predict_multiDEGGs for multiDEGGs_filter_combined()*

Description

This function generates predictions by creating a dataset with single and combined predictors based on the filtering results of a multiDEGGs_filter model.

Usage

```
## S3 method for class 'multiDEGGs_filter_combined'
predict(object, newdata, interaction.type = "ratio", sep = ":", ...)
```

Arguments

`object`	A fitted object of class `multiDEGGs_filter` containing filtering results with: **keep** Character vector of variable names to keep as single predictors **pairs** Data frame or matrix with two columns specifying pairs of variables to combine
`newdata`	A data frame containing the new data for prediction. Must contain all variables specified in `object$keep` and `object$pairs`.
`interaction.type`	Character string specifying how to combine the paired predictors. Options are: **"ratio"** Combine paired predictors by dividing the first variable by the second (a/b) **other** Combine paired predictors by multiplying the variables (a*b) Default is "ratio".
`sep`	Character string used as separator when creating column names for combined predictors. Default is ":".
`...`	Additional arguments passed to the generic function.

Details

The function processes the filtering results in two steps:

1. Selects single predictors from `newdata` based on variables listed in `object$keep`
2. Adds combined predictors from paired variables in `object$pairs`

Value

A data frame containing:

- Single predictors (if any are specified in `object$keep`)
- Combined predictors based on variable pairs and interaction type

---

`synthetic_metadata` *Synthetic clinical data*

Description

A dataset containing sample clinical data for 100 patients with 40% response rate

Format

A data frame with 100 rows and 4 columns (IDs are in rownames):

patient_id IDs matching the IDs used in the colnames of the assay data matrix/matrices.

age A column to simulate age of patients. Not used.

gender A column to simulate gender of patients. Not used.

response The response outcome, to be used for differential analysis

synthetic_OlinkData *Synthetic RNA-seq count data*

Description

Synthetic RNA-seq data after log2 normalisation

Format

A data frame with xx rows (proteins) xx columns (patients IDs).

synthetic_proteomicData *Synthetic RNA-seq count data*

Description

Synthetic RNA-seq data after log2 normalisation

Format

A data frame with xx rows (proteins) xx columns (patients IDs).

synthetic_rnaseqData *Synthetic RNA-seq count data*

Description

Synthetic RNA-seq data after log2 normalisation

Format

A data frame with xx rows (genes) xx columns (patients IDs, matching the metadata rownames).

tidy_metadata *Tidying up of metadata. Samples belonging to undesidered categories (if specified) will be removed as well as categories with less than five samples, and NAs.*

Description

Tidying up of metadata. Samples belonging to undesidered categories (if specified) will be removed as well as categories with less than five samples, and NAs.

Usage

```
tidy_metadata(
  category_subset = NULL,
  metadata,
  category_variable = NULL,
  verbose = FALSE
)
```

Arguments

category_subset	optional character vector indicating which categories are used for comparison. If not specified, all categories in `category_variable` will be used.
metadata	a named vector, matrix, or data.frame containing sample annotations or categories. If matrix or data.frame, each row should correspond to a sample, with columns representing different sample characteristics (e.g., treatment group, condition, time point). The colname of the sample characteristic to be used for differential analysis must be specified in `category_variable`. Rownames must match the sample IDs used in assayData. If named vector, each element must correspond to a sample characteristic to be used for differential analysis, and names must match sample IDs used in the colnames of `assayData`. Continuous variables are not allowed.
category_variable	column name in `metadata` (if data.frame or matrix) or NULL if `metadata` is already a named vector containing category information.
verbose	Logical. Whether to print detailed output messages during processing. Default is FALSE.

Value

a tidy named factor vector of sample annotations.

View_diffNetworks *Interactive visualisation of differential networks*

Description

Explore differential networks and interactively select regression and box plots

Usage

```
View_diffNetworks(deggs_object, legend.arrow.width = 0.35, stepY_legend = 55)
```

Arguments

deggs_object	an object of class deggs generated by get_diffNetworks
legend.arrow.width	width of the arrow used in the network legend. Default is 0.35. As the number of assayData matrices increases this parameter must be accordingly increased to avoid graphical errors in the legend.
stepY_legend	vertical space between legend arrows. It is used together with legend.arrow.width to adjust the legend space in case of graphical errors. Default is 55.

Value

a shiny interface showing networks with selectable nodes and links

Examples

```
data("synthetic_metadata")
data("synthetic_rnaseqData")
data("synthetic_proteomicData")
data("synthetic_OlinkData")
assayData_list <- list("RNAseq" = synthetic_rnaseqData,
                       "Proteomics" = synthetic_proteomicData,
                       "Olink" = synthetic_OlinkData)
deggs_object <- get_diffNetworks(assayData = assayData_list,
                                 metadata = synthetic_metadata,
                                 category_variable = "response",
                                 regression_method = "lm",
                                 verbose = FALSE,
                                 show_progressBar = FALSE,
                                 cores = 1)
# the below function runs a shiny app, so can't be run during R CMD check
if(interactive()){
View_diffNetworks(deggs_object)
}
```

Index