Package 'bmdx'

August 28, 2025

Title BMDx: Dose dependent analysis

Version 2.0

Description The BMDx R package offers a robust solution for Benchmark Dose (BMD) analysis of transcriptomics data. The package employs a sophisticated approach involving the fitting of diverse models and selecting the optimal one based on the Akaike Information Criterion (AIC) or model average. Key functionalities of BMDx include the computation of BMD, related values, and IC50/EC50 estimations.BMDx particularly excels in comparing BMD values across different time points within a transcriptomics experiment. BMDx is adept at handling and analyzing multiple experiments concurrently, enhancing the efficiency of dataset assessment and promoting informed decision-making through thorough data analysis.

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

LazyData true

Imports car,
dendextend,
dplyr,
drc,
ggplot2,
heatmaply,
investr,
limma,
modelr,
readxl,
trend,
viridis,
utils,
minpack.lm,
stats,
forcats,
plotly,
UpSetR,
igraph,
visNetwork,
networkD3,
scam

Contents

`add_average_models` *Add average models to a list of fitted models*

Description

This function adds average models to a list of fitted models based on the model averaging approach described in "A brief guide to model selection, multimodel inference and model averaging in behavioural ecology using Akaike's information criterion" by Matthew R. E. Symonds and Adnan Moussalli. The average model is computed using the provided models and added to the list under the name "average". The average model is fitted using the data from the first model in the list. The function returns the updated list of fitted models.

Usage

```
add_average_models(fitted_models, variance_type = "constant")
```

Arguments

fitted_models	The list of fitted models.
variance_type	variance type, possible values are: constant, nonconstant, infer.

Value

The updated list of fitted models with the average model added.

---

aggregate_rows_time *Aggregate Rows by Time and Other Features*

Description

Aggregates rows of a data frame based on time and other specified features.

Usage

```
aggregate_rows_time(
  mod_stats,
  gen_feat = "Feature",
  first_feat = "SACRI_PERIOD",
  group_by = c("Experiment", "DILI"),
  filter_column = NULL,
  filter_by = list(c("acetaminophen", "bucetin"))
)
```

Arguments

mod_stats	A data frame containing model statistics.
gen_feat	The general feature to aggregate.
first_feat	The first feature used in aggregation.
group_by	A character vector specifying the grouping features.
filter_column	A character vector specifying the columns to filter.
filter_by	A list of character vectors containing filter values for each column.

Value

Returns a ggplot2 plot of aggregated data or an empty plot if conditions are not met.

`apply_tpod_methods` *Apply tPOD Methods to Model Statistics*

Description

This function computes the tPOD (toxicological point of departure) using various methods and returns a data frame with the computed values. Errors during computation are handled gracefully, with NA returned for failed calculations.

Usage

```
apply_tpod_methods(
  model_stats,
  tpod_methods_list = c("percentile", "mean", "first_mode", "lowest", "accumulation"),
  pod_value = "BMD",
  percentile = 0.95,
  lowest_method = "lowest",
  bw_adjust = 3,
  robust = F,
  plot = F,
  ratio_threshold = 1.66,
  rank = 1
)
```

Arguments

`model_stats`	A data frame containing model statistics used for tPOD computation.
`tpod_methods_list`	Character vector; methods to apply, options include "percentile", "mean", "first_mode", "lowest", "accumulation".
`pod_value`	Character; type of point of departure to use: "BMD", "BMDL", or "BMDU". Default is "BMD".
`percentile`	Numeric; a value between 0 and 1 indicating the target percentile. Default is 0.95.
`lowest_method`	Character; method for selecting the lowest value: "lowest" or "LCRD". Default is "lowest".
`bw_adjust`	Numeric; bandwidth adjustment for density estimation methods. Default is 3.
`robust`	Logical; whether to use robust statistics where applicable. Default is FALSE.
`plot`	Logical; whether to generate plots during tPOD computation. Default is FALSE.
`ratio_threshold`	Numeric; threshold for ratio-based decisions. Default is 1.66.
`rank`	Numeric; rank selection for the tPOD if multiple candidates exist. Default is 1.

Value

A data frame containing the computed tPOD values with columns: "Method", "Parameter", and "tPOD".

bmr_factor *Calculate the benchmark response level*

Description

This function calculates the benchmark response level based on the given risk factor, whether the response should be increased or decreased, and the background level. The calculation is based on Equation 14 from the paper referenced in the code.

Usage

```
bmr_factor(risk_factor = 0.1, increase = TRUE, background_level = 0.01)
```

Arguments

risk_factor	The risk factor used to determine the benchmark response level. Default is 0.1.
increase	Logical value indicating whether the response should be increased. If TRUE, the response is increased; if FALSE, it is decreased. Default is TRUE.
background_level	The background level of the response. Default is 0.01.

Value

The benchmark response level calculated based on the input parameters.

build.model.matrix *Build Model Matrix*

Description

This function constructs the design matrix for a linear model based on the provided phenotype data, variable of interest, and optional covariates.

Usage

```
build.model.matrix(
  pd,
  intercept = -1,
  var.int,
  covariates = NULL,
  verbose = TRUE
)
```

Arguments

pd	The phenotype data as a data frame.
intercept	The value to be used for the intercept (default: -1).
var.int	The variable of interest used in the design matrix.
covariates	A character vector specifying the optional covariates to include in the design matrix (default: NULL).
verbose	Logical; if TRUE, print the formula used for model.matrix (default: TRUE).

Value

The design matrix for the linear model.

`build_models` *Build multiple models based on the given model names.*

Description

This function constructs a list of model objects based on the specified model names. The available model families include "linear", "hill", "power", "poly2", "poly3", "poly4", "poly5", "exp2", "exp3", "exp4", "exp5", "llog2", "llog3", "llog4", "llog5", "mm2", "weibul12", "weibul13", "weibul14", "weibul22", "weibul23", and "weibul24".

Usage

```
build_models(
 model_names = c("linear", "hill", "power", "poly2", "poly3", "poly4", "poly5", "exp2",
   "exp3", "exp4", "exp5", "llog2", "llog3", "llog4", "llog5", "mm2", "weibul12",
    "weibul13", "weibul14", "weibul22", "weibul23", "weibul24"),
 max_iter = 1024,
 data_type = c("continuous", "binomial"),
 x,
 y
)
```

Arguments

`model_names`	A character vector specifying the model names to build. Default is all available model names.
`max_iter`	Maximum number of iterations for iterative model fitting. Default is 1024.
`data_type`	Type of data, either "continuous" or "binomial". Default is "continuous".
`x`	The predictor variable (independent variable).
`y`	The response variable (dependent variable).

Value

model_list_result A list containing the specified model objects.

calculate_lcrd *Calculate the Lowest Consistent Response Dose (LCRD)*

Description

This function identifies the lowest consistent response dose (LCRD) from a numeric vector of BMC values. The BMCs are ranked in ascending order, and a consistency check is applied such that the ratio between consecutive BMCs (BMCn+1 / BMCn) must be < 1.66. The first BMC for which all subsequent ratios remain below this threshold is declared the LCRD.

Usage

```
calculate_lcrd(pod_vector, ratio_threshold)
```

Arguments

pod_vector	A numeric vector of BMC values.
ratio_threshold	The maximum allowed ratio between consecutive BMCs (default: 1.66).

Value

A list containing the LCRD, the CRGB (consistent response group of BMCs), and the ranked BMCs.

check_edges_in_graph *Check Presence of Edges in a Graph*

Description

This function checks whether edges in a data frame are present in the specified `igraph` object. It uses the `filter_existing_nodes()` helper to ensure only valid nodes are checked, and then marks presence via a logical column.

Usage

```
check_edges_in_graph(
  graph,
  edge_df,
  feature1 = "Feature 1",
  feature2 = "Feature 2"
)
```

Arguments

graph	An `igraph` object representing the graph to check against.
edge_df	A data frame where the first two columns are node names for the edges to be checked.

Value

A data frame with an additional logical column indicating whether each edge is present in the graph.

`cluster_genes_pairs` *Cluster Gene Pairs Based on Expression Profile Similarity*

Description

This function clusters gene pairs using similarity metrics derived from pairwise comparisons (e.g., correlation and distance). It constructs a matrix representing gene-gene similarity, performs hierarchical clustering, and generates a `ComplexHeatmap` annotated with cluster and centrality information. Optionally, it overlays protein-protein interaction (PPI) data to highlight known gene interactions.

Usage

```
cluster_genes_pairs(
  comparison_pairs,
  nclust = 2,
  method = "euclidean",
  plot_ppi_info = TRUE,
  gene_id_type = "ENSEMBL",
  organism = "human"
)
```

Arguments

`comparison_pairs`	A data frame of pairwise gene comparisons. Must contain columns `"Feature 1"`, `"Feature 2"`, `"CorGenePatteerns"` (correlation), and `"NormalizedEuclideanDistance"` (distance).
`nclust`	Integer. Number of clusters to extract from hierarchical clustering. Default is 2.
`method`	A string specifying the clustering metric: `"euclidean"`, `"correlation"`, or `"combination"`. If `"combination"`, both normalized distance and inverse correlation are averaged.
`plot_ppi_info`	Logical. If `TRUE`, adds PPI network information and node centrality measures (degree, closeness, eigenvector) to the heatmap annotations. Default is `TRUE`.
`gene_id_type`	A string indicating the type of gene identifier used (e.g., `"ENSEMBL"`, `"ENTREZ"`, `"SYMBOL"`).
`organism`	A string specifying the organism: `"human"`, `"mouse"`, or `"rat"`.

Details

The function builds symmetric matrices of pairwise correlation and Euclidean distance, performs hierarchical clustering on the combined or selected metric, and annotates the heatmap with cluster statistics and optional PPI features. When `plot_ppi_info = TRUE`, it highlights interactions using dot markers and colors nodes based on centrality measures derived from organism-specific PPI graphs.

Value

A list with the following elements:

clusters A named vector of cluster assignments for each gene.

n_genes_par_cluster A table of gene counts per cluster.

average_correlation_in_clusters A numeric vector of mean intra-cluster similarity values.

list_correlation_in_clusters A list of submatrices showing correlation values within each cluster.

complex_heatmap A `ComplexHeatmap` object visualizing the clustered similarity matrix.

`compute_deviations` *Compute deviations for BMD modeling*

Description

This function computes deviations for BMD modeling based on the specified deviation type. Three different types of deviation are available: standard deviation, relative deviation, and absolute deviation. Refer to the EPA documentation for detailed information on the deviation types (slide 7) from the provided link.

Usage

```
compute_deviations(deviation_type = "standard", model, rl = 1.349)
```

Arguments

`deviation_type`	Character string specifying the type of deviation to compute. Possible values are "standard", "relative", and "absolute".
`model`	The model object used for computation.
`rl`	The relative level used in the deviation calculation. Default is 1.349.

Value

The computed deviation value based on the specified deviation type.

`compute_gene_frequencies`
Compute Gene Frequencies and Create Lollipop Plots

Description

This function computes gene frequencies based on the provided model statistics and generates lollipop plots for the top genes based on their frequencies.

Usage

```
compute_gene_frequencies(
  mod_stats,
  th = 0.7,
  top_genes = 100,
  rel_variable = "Experiment",
  group_by = "None",
  split_by = "None"
)
```

Arguments

mod_stats	A data frame containing the model statistics and gene information.
th	Threshold value for gene percentage. Only genes with a percentage above this threshold will be plotted.
top_genes	Maximum number of genes to plot. Default = 100
rel_variable	The name of the variable representing the experiments or conditions.
group_by	The name of the variable to group the data for generating lollipop plots.
split_by	The name of the variable to split the data and generate separate lollipop plots.

Value

A list containing gene lists, lollipop plots, and matrices for each split-by value or for all data.

---

compute_model_statistics

Compute model statistics for fitted models

Description

This function computes model statistics for a list of fitted models and returns the results as a data frame.

Usage

```
compute_model_statistics(
  fitted_models,
  other_variables_id_col,
  is_parallel = TRUE,
  nCores = 2
)
```

Arguments

fitted_models	A list of fitted models.
other_variables_id_col	The name of the column representing the other variables in the model statistics data frame.
is_parallel	Whether to compute the statistics in parallel. Default is TRUE.
nCores	The number of cores to use for parallel computation. Default is 2.

Value

A data frame containing the computed model statistics.

`create_data_structure` *this function convert the data in input into the format required for dose-dependent modelling*

Description

this function convert the data in input into the format required for dose-dependent modelling

Usage

```
create_data_structure(
  experimental_data,
  metadata,
  sample_id_col = "BARCODE",
  dose_id_col = "DOSE",
  other_variables_id_col = c("SACRI_PERIOD"),
  x = "dose",
  y = "expr"
)
```

Arguments

`experimental_data`	a list of dataframe containing experimental data. Each row is a feature (e.g. gene) and each column is a sample
`metadata`	a list of dataframe containing the metadata for the experimental data. Each row is a sample and the columns represent the different variables. A column for dose/concentration is required
`sample_id_col`	a character specifying the name of the column containing the samples id
`dose_id_col`	a character specifying the name of the column containing the doses/concentration
`other_variables_id_col`	a vector of characters specifying the name of the column used to group the data
`x`	a characters specifying the name of the x variable in the model. Default is dose.
`y`	a characters specifying the name of the y variable in the model. Default is expr.

Value

a dictionary containing the data frame for modelling. Dictionary Keys are n-uple specifying the experiment name, other variables of interests and feature names (e.g. drug, time, gene)

`diff.gene.expr` *Differential Gene Expression Analysis*

Description

This function performs differential gene expression analysis using limma's linear model with specified contrasts and adjustment method.

Usage

```
## S3 method for class 'gene.expr'
diff(data, des, contrasts, adjust.method)
```

Arguments

`data`	The gene expression data as a data frame or matrix.
`des`	The design matrix representing the experimental design.
`contrasts`	A character vector specifying the contrasts for analysis.
`adjust.method`	The method for p-value adjustment (default: "none"). Options are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none".

Value

A list containing topTable results for each contrast specified.

`dose_response_analysis` *Perform dose-response analysis on a list of models*

Description

This function fits a list of models to the same end-point and performs dose-response analysis, including estimation of BMD, BMDL, BMDU, and AC50 values.

Usage

```
dose_response_analysis(
  data,
  model_list,
  deviation_type = "standard",
  rl = 1.349,
  variance_type = "constant",
  confidence_interval = 0.95,
  significance_level = 0.05
)
```

Arguments

`data`	The data used for fitting the models.
`model_list`	A list of models to fit to the data.
`deviation_type`	Character string specifying the type of deviation from the fitted model to use for BMD calculation. Default is "standard". Allowed values are standard and relative
`rl`	The relative level used to calculate the BMD. Default is 1.349.
`variance_type`	Character string specifying the type of variance to use in model fitting. Default is "constant". Other possible values are "non constant", "model" and "inferred"
`confidence_interval`	The confidence level for the confidence interval. Default is 0.95.
`significance_level`	The significance level for model fitting. Default is 0.05.

Value

A list of models with additional attributes for BMDL, BMDU, and AC50 values. Models that fail to fit or encounter an error during estimation are excluded from the returned list.

`ecdf_plots` *Make ECDF or Histogram Plots for BMD Data*

Description

This function creates ECDF (Empirical Cumulative Distribution Function) or Histogram plots for the BMD (Benchmark Dose) data in the provided model statistics data frame.

Usage

```
ecdf_plots(
  mod_stats,
  rel_variable = "Experiment",
  group_by = NULL,
  is_group_by_numeric = TRUE,
  other_variables = NULL,
  number_of_column = 2,
  scaling = TRUE,
  filter_column = c("Model"),
  filter_by = list(c("linear")),
  plot_type = "ecdf",
  linewidth = 1.3,
  text_size = 14
)
```

Arguments

mod_stats	The model statistics data frame.
rel_variable	The column name in mod_stats representing the experimental condition or treatment (default: "Experiment").
group_by	The column name in mod_stats used to group the data for faceting (default: NULL).
is_group_by_numeric	Logical; if TRUE, treat the group_by variable as numeric (default: TRUE).
other_variables	A character vector specifying additional variables for faceting (default: NULL).
number_of_column	The number of columns in the facet grid (default: 2).
scaling	Logical; if TRUE, scale the BMD values (default: TRUE).
filter_column	The column name(s) in mod_stats to use for filtering (default: "Model").
filter_by	A list of character vectors containing filtering criteria for each filter_column (default: list(c("linear"))).
plot_type	The type of plot to create; either "ecdf" for ECDF or "histogram" for histogram (default: "ecdf").
linewidth	Integer specifing the width of the lines. Default 1.3
text_size	Integer specifing the font size. Default = 14

Value

A plotly object representing the ECDF or histogram plot.

---

ens_human_gene_graph *ens_human_gene_graph*

Description

Graph whose nodes are Human genes (Ensembl ID). Edges represent protein-protein interactions (PPIs).

Usage

```
ens_human_gene_graph
```

Format

An object of class igraph of length 20817.

Source

https://doi.org/10.1093/bib/bbaa417

ens_human_gene_graph_mirna *ens_human_gene_graph_mirna*

Description

Graph whose nodes are Human genes (Ensembl ID). Edges represent regulatory interactions between miRNA and genes.

Usage

```
ens_human_gene_graph_mirna
```

Format

An object of class igraph of length 10662.

Source

https://doi.org/10.1093/bib/bbaa417

ens_human_gene_graph_tf *ens_human_gene_graph_tf*

Description

Graph whose nodes are Human genes (Ensembl ID). Edges represent regulatory interactions between transcription factor (TF) and genes.

Usage

```
ens_human_gene_graph_tf
```

Format

An object of class igraph of length 18753.

Source

https://doi.org/10.1093/bib/bbaa417

ens_mouse_gene_graph *ens_mouse_gene_graph*

Description

Graph whose nodes are Mouse genes (Ensembl ID). Edges represent protein-protein interactions (PPIs).

Usage

```
ens_mouse_gene_graph
```

Format

An object of class igraph of length 19323.

Source

https://doi.org/10.1093/bib/bbaa417

ens_mouse_gene_graph_mirna *ens_mouse_gene_graph_mirna*

Description

Graph whose nodes are Mouse genes (Ensembl ID). Edges represent regulatory interactions between miRNA and genes.

Usage

```
ens_mouse_gene_graph_mirna
```

Format

An object of class igraph of length 4294.

Source

https://doi.org/10.1093/bib/bbaa417

ens_mouse_gene_graph_tf *ens_mouse_gene_graph_tf*

Description

Graph whose nodes are Mouse genes (Ensembl ID). Edges represent regulatory interactions between transcription factor (TF) and genes.

Usage

```
ens_mouse_gene_graph_tf
```

Format

An object of class `igraph` of length 6489.

Source

https://doi.org/10.1093/bib/bbaa417

ens_rat_gene_graph *ens_rat_gene_graph*

Description

Graph whose nodes are Rat genes (Ensembl ID). Edges represent protein-protein interactions (PPIs).

Usage

```
ens_rat_gene_graph
```

Format

An object of class `igraph` of length 17140.

Source

https://doi.org/10.1093/bib/bbaa417

ens_rat_gene_graph_mirna *ens_rat_gene_graph_mirna*

Description

Graph whose nodes are Rat genes (Ensembl ID). Edges represent regulatory interactions between miRNA and genes.

Usage

```
ens_rat_gene_graph_mirna
```

Format

An object of class `igraph` of length 36.

Source

https://doi.org/10.1093/bib/bbaa417

ens_rat_gene_graph_tf *ens_rat_gene_graph_tf*

Description

Graph whose nodes are Rat genes (Ensembl ID). Edges represent regulatory interactions between transcription factor (TF) and genes.

Usage

```
ens_rat_gene_graph_tf
```

Format

An object of class `igraph` of length 172.

Source

https://doi.org/10.1093/bib/bbaa417

ent_human_gene_graph *ent_human_gene_graph*

Description

Graph whose nodes are Human genes (Entrez ID). Edges represent protein-protein interactions (PPIs).

Usage

```
ent_human_gene_graph
```

Format

An object of class igraph of length 120356739.

Source

https://doi.org/10.1093/bib/bbaa417

ent_human_gene_graph_mirna *ent_human_gene_graph_mirna*

Description

Graph whose nodes are Human genes (Entrez ID). Edges represent regulatory interactions between miRNA and genes.

Usage

```
ent_human_gene_graph_mirna
```

Format

An object of class igraph of length 120356739.

Source

https://doi.org/10.1093/bib/bbaa417

ent_human_gene_graph_tf *ent_human_gene_graph_tf*

Description

Graph whose nodes are Human genes (Entrez ID). Edges represent regulatory interactions between transcription factor (TF) and genes.

Usage

```
ent_human_gene_graph_tf
```

Format

An object of class igraph of length 120356739.

Source

https://doi.org/10.1093/bib/bbaa417

ent_mouse_gene_graph *ent_mouse_gene_graph*

Description

Graph whose nodes are Mouse genes (Entrez ID). Edges represent protein-protein interactions (PPIs).

Usage

```
ent_mouse_gene_graph
```

Format

An object of class igraph of length 118568460.

Source

https://doi.org/10.1093/bib/bbaa417

ent_mouse_gene_graph_mirna *ent_mouse_gene_graph_mirna*

Description

Graph whose nodes are Mouse genes (Entrez ID). Edges represent regulatory interactions between miRNA and genes.

Usage

```
ent_mouse_gene_graph_mirna
```

Format

An object of class `igraph` of length 102466886.

Source

https://doi.org/10.1093/bib/bbaa417

ent_mouse_gene_graph_tf *ent_mouse_gene_graph_tf*

Description

Graph whose nodes are Mouse genes (Entrez ID). Edges represent regulatory interactions between transcription factor (TF) and genes.

Usage

```
ent_mouse_gene_graph_tf
```

Format

An object of class `igraph` of length 102467006.

Source

https://doi.org/10.1093/bib/bbaa417

ent_rat_gene_graph *ent_rat_gene_graph*

Description

Graph whose nodes are Rat genes (Entrez ID). Edges represent protein-protein interactions (PPIs).

Usage

```
ent_rat_gene_graph
```

Format

An object of class igraph of length 120102993.

Source

https://doi.org/10.1093/bib/bbaa417

ent_rat_gene_graph_mirna *ent_rat_gene_graph_mirna*

Description

Graph whose nodes are Rat genes (Entrez ID). Edges represent regulatory interactions between miRNA and genes.

Usage

```
ent_rat_gene_graph_mirna
```

Format

An object of class igraph of length 104796155.

Source

https://doi.org/10.1093/bib/bbaa417

ent_rat_gene_graph_tf *ent_rat_gene_graph_tf*

Description

Graph whose nodes are Rat genes (Entrez ID). Edges represent regulatory interactions between transcription factor (TF) and genes.

Usage

```
ent_rat_gene_graph_tf
```

Format

An object of class `igraph` of length 114483548.

Source

https://doi.org/10.1093/bib/bbaa417

filter_antimode *Filter Data Based on Antimode of BMD Distribution*

Description

This function filters `optimal_models_stats` based on the **antimode** of the BMD distribution. It splits the data into two groups:

- **Before Antimode**: BMD values **less than or equal to** the antimode.
- **After Antimode**: BMD values **greater than** the antimode but **below the 95th percentile**.

Usage

```
filter_antimode(optimal_models_stats, timepoints_col_variable)
```

Arguments

`optimal_models_stats`
: A dataframe containing the BMD values and corresponding timepoints.

`timepoints_col_variable`
: A character vector specifying the timepoints column variable to analyze.

Value

A **list** with two elements:

- **$before**: Dataframe containing BMD values **before** the antimode.
- **$after**: Dataframe containing BMD values **after** the antimode but below the 95th percentile.

filter_df *this function filters the dataframe of the models statistic*

Description

this function filters the dataframe of the models statistic

Usage

```
filter_df(mod_stats, filter_column, filter_by)
```

Arguments

mod_stats	the dataframe containing all the statistics
filter_column	a vector of dataframe columns to be used for filtering
filter_by	a list containing the selected values to be mantained after filtering. Each position of the list correspond to one of the selected column and contains a vector of admissible values #The length of filter_column and filter_by must be the same

Value

the filtered dataframe

filter_df_no *Filter DataFrame by Column Values*

Description

Filters a data frame based on specified columns and filter values.

Usage

```
filter_df_no(mod_stats, filter_column, filter_by)
```

Arguments

mod_stats	A data frame containing model statistics.
filter_column	A character vector specifying the columns to filter.
filter_by	A list of character vectors containing filter values for each column.

Value

Returns a filtered data frame if filtering conditions are met; otherwise, NULL.

filter_existing_nodes *Filter Edges Between Existing Nodes in a Graph*

Description

This function filters an edge data frame to include only those edges for which both nodes exist in the provided igraph object.

Usage

```
filter_existing_nodes(
  graph,
  edge_df,
  feature1 = "Feature 1",
  feature2 = "Feature 2"
)
```

Arguments

graph	An igraph object containing the network structure.
edge_df	A data frame where the first two columns represent source and target node names, respectively.

Value

A filtered data frame containing only the edges for which both nodes exist in the graph.

filter_pval *Filter results based on p-value threshold*

Description

This function filters the results based on a specified p-value threshold. It returns the filtered ANOVA results and the indices of the filtered rows.

Usage

```
filter_pval(filtering_res, Pval.th = 0.05)
```

Arguments

filtering_res	The filtering results table.
Pval.th	The p-value threshold for filtering the results. Default is 0.05.

Value

A list containing the filtered results and the indices of the filtered rows.

filter_pval_th *Filter Fold Change DataFrame by p-value and fold change threshold*

Description

This function filters a fold change data frame based on given p-value and fold change thresholds.

Usage

```
filter_pval_th(fold_change_dataframe, fcPval.th = 0.05, fc.th = 0.58)
```

Arguments

fold_change_dataframe	A data frame containing fold change results.
fcPval.th	The p-value threshold for filtering (default: 0.05).
fc.th	The fold change threshold for filtering (default: 0.58).

Value

A numeric vector containing the indices of rows that pass the filtering criteria.

filter_tpod_acc *Filter Data Based on tpod accumulation for Given Timepoints*

Description

This function filters data based on the BMD values for specified timepoints. It applies the `tpod_accumulation` function to determine the antimode and then divides the data into two groups:

- `before`: data where BMD values are less than or equal to the antimode.
- `after`: data where BMD values are greater than the antimode but less than the 95th percentile.

Usage

```
filter_tpod_acc(optimal_models_stats, timepoints_col_variable)
```

Arguments

optimal_models_stats	A data frame containing the columns `timepoint` and `BMD`. This data frame holds the information about the optimal models for different timepoints.
timepoints_col_variable	A character vector specifying the timepoints column variable to analyze.
plot	A logical value indicating whether to generate a plot of the accumulation POD distribution with a vertical line showing the computed tPOD. Default is `FALSE`.

Value

A list containing two data frames: - `before`: data frame with entries where BMD values are less than or equal to the antimode. - `after`: data frame with entries where BMD values are greater than the antimode but less than the 95th percentile.

`find_best_model_aic` *Find the best model based on AIC (Akaike Information Criterion) This function finds the best model based on the Akaike Information Criterion (AIC) among a list of models.*

Description

Find the best model based on AIC (Akaike Information Criterion) This function finds the best model based on the Akaike Information Criterion (AIC) among a list of models.

Usage

```
find_best_model_aic(model_list)
```

Arguments

`model_list`	A list containing multiple models to be compared.

Value

The model from the list that has the lowest AIC value.

`fitting_list` *Fit dose-response models to data dictionary*

Description

This function fits dose-response models to a data dictionary using the specified model list. It returns a list of fitted models.

Usage

```
fitting_list(
  data_dictionary,
  model_list,
  deviation_type = "relative",
  rl = 1.349,
  confidence_interval = 0.95,
  variance_type = "constant",
  significance_level = 0.05,
  is_parallel = FALSE,
  nCores = 2
)
```

Arguments

data_dictionary	The data dictionary containing the data frames to fit the models to.
model_list	A list of dose-response models to fit.
deviation_type	The type of deviation to use for BMD calculations. Default is "relative".
rl	The constant value for relative deviation. Default is 1.349.
confidence_interval	The confidence level for the BMD calculation. Default is 0.95.
variance_type	The type of variance to assume for the models. Default is "constant".
significance_level	The significance level for the BMD calculation. Default is 0.05.
is_parallel	Whether to perform the fitting in parallel. Default is FALSE.
nCores	The number of cores to use for parallel computation. Default is 2.

Value

A list containing the fitted models.

---

gene_pairs_analysis *Perform gene pairs analysis for selected experiments and times This function performs gene pairs analysis for selected experiments and times, using the provided filtered_optimal_models, BMD_tab, length_vectors, nCores, phenoList, doseColID, timeColID, and other_variables_id_col.*

Description

Perform gene pairs analysis for selected experiments and times This function performs gene pairs analysis for selected experiments and times, using the provided filtered_optimal_models, BMD_tab, length_vectors, nCores, phenoList, doseColID, timeColID, and other_variables_id_col.

Usage

```
gene_pairs_analysis(
  select_experiment,
  select_time,
  filtered_optimal_models,
  BMD_tab,
  length_vectors,
  nCores,
  phenoList,
  doseColID,
  timeColID,
  other_variables_id_col
)
```

Arguments

select_experiment	A character vector specifying the selected experiments for the analysis.
select_time	A numeric vector specifying the selected times for the analysis.
filtered_optimal_models	A list of filtered optimal models.
BMD_tab	The BMD (Benchmark Dose) table.
length_vectors	An integer specifying the length of vectors.
nCores	An integer specifying the number of cores to use for parallel processing.
phenoList	A list containing the phenotype data.
doseColID	The column ID for the dose in the phenotype data.
timeColID	The column ID for the time in the phenotype data.
other_variables_id_col	The column ID for other variables in the phenotype data.

Value

A list containing gene pairs statistics and the newdata used for the analysis.

---

gene_pairs_comparison *Compare gene pairs for their dose-dependent patterns This function compares gene pairs for their dose-dependent patterns using the provided filtered_optimal_models, other_variables_id_col, and newdata. It calculates the difference, behavior, correlation, and coefficient for each pair of genes in the models.*

Description

Compare gene pairs for their dose-dependent patterns This function compares gene pairs for their dose-dependent patterns using the provided filtered_optimal_models, other_variables_id_col, and newdata. It calculates the difference, behavior, correlation, and coefficient for each pair of genes in the models.

Usage

```
gene_pairs_comparison(
  filtered_optimal_models,
  other_variables_id_col,
  newdata,
  nCores = 40
)
```

Arguments

filtered_optimal_models	A list of filtered optimal models containing gene-related information.
other_variables_id_col	The column ID for other variables in the gene data.
newdata	A data frame containing dose information for new observations.
nCores	An integer specifying the number of cores to use for parallel processing (default is 40).

Value

A data frame containing gene pairs statistics, including Experiment 1, Model 1, Experiment 2, Model 2, Difference Trend, Coefficient, and Correlation of Gene Patterns.

`get_model_stats` *given a model the function creates a named vector with all the model statistics*

Description

given a model the function creates a named vector with all the model statistics

Usage

```
get_model_stats(model)
```

Arguments

`model`	an object of class bmdx

Value

model_stats a named vector with statistics

`get_mode_antimode` *Identify the first two local maxima and the minimum ("valley") between them in a density estimate*

Description

This function estimates the density of a numeric vector `x`, identifies the first two local maxima (peaks), and locates the lowest point (valley) between those peaks. It returns the x-values for the first peak (mode), the valley (antimode), and a function for interpolating the estimated PDF.

Usage

```
get_mode_antimode(x, bw_adjust = 3)
```

Arguments

`x`	A numeric vector from which to estimate a density.
`bw_adjust`	A numeric multiplier for the bandwidth. The bandwidth is chosen via the "SJ" method and then multiplied by `bw_adjust`.

Value

A list with elements:

mode The x-value of the first local maximum (peak).

antimode The x-value of the local minimum (valley) between the first two peaks.

pdf A function that interpolates (and extrapolates) the estimated density at arbitrary points.

`inner.check.model` *Check if a model passes the filtering criteria*

Description

This function checks if a model passes the specified filtering criteria. The criteria include thresholds, lack of fit, ratios, missing values, R-squared, and monotonicity. The function returns TRUE if the model passes all the criteria, and FALSE otherwise.

Usage

```
inner.check.model(
  mod,
  loofth = 0.1,
  lower_bound_th = 0.1,
  upper_bound_th = 0.1,
  bmd_bmdl_th = 20,
  bmdu_bmd_th = 20,
  bmdu_bmdl_th = 40,
  filter_lower_bound = TRUE,
  filter_upper_bound = TRUE,
  filter_by_lack_of_fit = TRUE,
  ratio_filter = TRUE,
  bmd_na_filter = TRUE,
  bmdl_na_filter = TRUE,
  bmdu_na_filter = TRUE,
  ic50_na_filter = TRUE,
  r2_filter = FALSE,
  r2_th = 0.6,
  filter_by_monotonicity = FALSE,
  filter_by_negative_values = TRUE,
  filter_by_unordered_values = TRUE
)
```

Arguments

`mod`	The model to be checked.
`loofth`	The threshold for lack of fit. Default is 0.1.
`lower_bound_th`	The lower bound threshold. Default is 0.1% of the lowest dose.
`upper_bound_th`	The upper bound threshold. Default is 0.1% of the highest dose.
`bmd_bmdl_th`	The threshold for the ratio of BMD to BMDL. Default is 20.
`bmdu_bmd_th`	The threshold for the ratio of BMDU to BMD. Default is 20.
`bmdu_bmdl_th`	The threshold for the ratio of BMDU to BMDL. Default is 40.
`filter_lower_bound`	Logical value indicating whether to filter the model based on the lower bound threshold. Default is TRUE.
`filter_upper_bound`	Logical value indicating whether to filter the model based on the upper bound threshold. Default is TRUE.

`filter_by_lack_of_fit`
: Logical value indicating whether to filter the model based on lack of fit. Default is TRUE.

`ratio_filter`
: Logical value indicating whether to filter the model based on ratios. Default is TRUE.

`bmd_na_filter`
: Logical value indicating whether to filter the model with missing BMD values. Default is TRUE.

`bmdl_na_filter`
: Logical value indicating whether to filter the model with missing BMDL values. Default is TRUE.

`bmdu_na_filter`
: Logical value indicating whether to filter the model with missing BMDU values. Default is TRUE.

`ic50_na_filter`
: Logical value indicating whether to filter the model with missing IC50 values. Default is TRUE.

`r2_filter`
: Logical value indicating whether to filter the model based on R-squared. Default is FALSE.

`r2_th`
: The threshold for R-squared. Default is 0.6.

`filter_by_monotonicity`
: Logical value indicating whether to filter the model based on monotonicity. Default is FALSE.

Value

TRUE if the model passes the filtering criteria, FALSE otherwise.

`is_strictly_monotonic` *Check if a fitted model is strictly monotonic*

Description

This function checks if a fitted model exhibits strict monotonicity over the range of doses. It returns TRUE if the model is strictly monotonic and FALSE otherwise.

Usage

```
is_strictly_monotonic(fittedModel)
```

Arguments

`fittedModel`
: The fitted model to be checked.

Value

TRUE if the model is strictly monotonic, FALSE otherwise.

kneedle — *Identify the knee point in two vectors using a simple Kneedle-like approach*

Description

This function normalizes the input vectors `x` and `y` to the range [0, 1] and calculates the difference (`y_norm - x_norm`) at each point. The maximum of this difference is used as the index of the knee point, and the corresponding values of `x` and `y` at this index are returned.

Usage

```
kneedle(x, y)
```

Arguments

`x`	A numeric vector representing the x-coordinates.
`y`	A numeric vector representing the y-coordinates.

Details

The vectors `x` and `y` are first normalized to the [0, 1] range. The function then computes the difference (`y_norm - x_norm`) at each corresponding index and identifies the knee point as the location where this difference is maximized.

Value

A list with the following elements:

`index` The index of the knee point in the original vectors.

`x` The x-value at the knee point.

`y` The y-value at the knee point.

load_ppi_data — *Load Protein-Protein Interaction (PPI) Graph for an Organism*

Description

This function loads a precompiled protein-protein interaction (PPI) network graph for a specified organism and gene identifier type (e.g., ENSEMBL, ENTREZ, or SYMBOL). It returns the corresponding `igraph` object representing the network.

Usage

```
load_ppi_data(organism, gene_id_type, graph_type = "ppi")
```

Arguments

organism	A string indicating the organism name. Supported values are "human", "mouse", and "rat".
gene_id_type	A string indicating the gene identifier type. Supported values are "ENSEMBL", "ENTREZ", and "SYMBOL".
graph_type	A string indicating the graph type. Supported values are "ppi", and "tf".

Details

This function expects that the relevant data objects (e.g., ens_human_gene_graph, sym_mouse_gene_graph, etc.) are available in the environment or included as package data. The function uses data() to load the correct graph into the environment before returning it.

Value

An igraph object containing the PPI network for the specified organism and gene ID type.

---

make_d3_network *Create a D3 Network Visualization*

Description

This function generates a D3 network visualization based on statistical data and a graph structure.

Usage

```
make_d3_network(statistics, g, th = 0, positive = TRUE)
```

Arguments

statistics	A data frame containing the statistics on the gene-gene correlation data.
g	An igraph graph object representing gene-gene interactions.
th	A threshold for filtering correlations.
positive	Logical, whether to consider positive correlations.

Value

A list containing the D3 network visualization, a data frame of node statistics, and an igraph graph object.

make_empty_plot *Create Empty ggplot2 Plot Creates an empty ggplot2 plot with a void theme and no x-label.*

Description

Create Empty ggplot2 Plot Creates an empty ggplot2 plot with a void theme and no x-label.

Usage

```
make_empty_plot()
```

Value

Returns an empty ggplot2 plot.

model_filtering *Filter a list of fitted models based on various criteria*

Description

This function filters a list of fitted models based on specified criteria, such as thresholds, lack of fit, ratios, missing values, R-squared, and monotonicity. The function returns a filtered list of models that pass the specified criteria.

Usage

```
model_filtering(
  fitted_models,
  loofth = 0.1,
  lower_bound_th = 0.1,
  upper_bound_th = 0.1,
  bmd_bmdl_th = 20,
  bmdu_bmd_th = 20,
  bmdu_bmdl_th = 40,
  filter_lower_bound = TRUE,
  filter_upper_bound = TRUE,
  filter_by_lack_of_fit = TRUE,
  ratio_filter = TRUE,
  bmd_na_filter = TRUE,
  bmdl_na_filter = TRUE,
  bmdu_na_filter = TRUE,
  ic50_na_filter = TRUE,
  r2_filter = FALSE,
  r2_th = 0.6,
  filter_by_monotonicity = FALSE,
  filter_by_negative_values = TRUE,
  filter_by_unordered_values = TRUE
)
```

Arguments

fitted_models	A list of fitted models to be filtered.
loofth	The threshold for lack of fit. Default is 0.1.
lower_bound_th	The lower bound threshold. Default is 0.1% of the lowest dose.
upper_bound_th	The upper bound threshold. Default is 0.1% of the highest dose.
bmd_bmdl_th	The threshold for the ratio of BMD to BMDL. Default is 20.
bmdu_bmd_th	The threshold for the ratio of BMDU to BMD. Default is 20.
bmdu_bmdl_th	The threshold for the ratio of BMDU to BMDL. Default is 40.
filter_lower_bound	Logical value indicating whether to filter models based on the lower bound threshold. Default is TRUE.
filter_upper_bound	Logical value indicating whether to filter models based on the upper bound threshold. Default is TRUE.
filter_by_lack_of_fit	Logical value indicating whether to filter models based on lack of fit. Default is TRUE.
ratio_filter	Logical value indicating whether to filter models based on ratios. Default is TRUE.
bmd_na_filter	Logical value indicating whether to filter models with missing BMD values. Default is TRUE.
bmdl_na_filter	Logical value indicating whether to filter models with missing BMDL values. Default is TRUE.
bmdu_na_filter	Logical value indicating whether to filter models with missing BMDU values. Default is TRUE.
ic50_na_filter	Logical value indicating whether to filter models with missing IC50 values. Default is TRUE.
r2_filter	Logical value indicating whether to filter models based on R-squared. Default is FALSE.
r2_th	The threshold for R-squared. Default is 0.6.
filter_by_monotonicity	Logical value indicating whether to filter models based on monotonicity. Default is FALSE.
filter_by_negative_values	Logical value indicating wheather to filter models based on negative estimated effective doses
filter_by_unordered_values	Logival value indicating wheather to filter models for which BMDL<BMD<BMDU is not true

Value

A filtered list of models that pass the specified criteria.

`perform_anova` *Perform ANOVA on data dictionary*

Description

This function performs ANOVA on a data dictionary, computes p-values, performs adjustment if specified, filters the results based on a significance threshold, and returns the ANOVA results and filtered data dictionary.

Usage

```
perform_anova(
  data_dictionary,
  anovaAdjustment = "Nominal",
  anovaPval.th = 0.05,
  anovaCores = 1,
  x = "dose",
  y = "expr",
  other_variables_id_col = NULL
)
```

Arguments

`data_dictionary`	The data dictionary containing the data frames to perform ANOVA on.
`anovaAdjustment`	The adjustment method for p-values. Options include "Nominal" (no adjustment) and methods supported by the `p.adjust` function. Default is "Nominal".
`anovaPval.th`	The significance threshold for filtering the ANOVA results. Default is 0.05.
`anovaCores`	The number of cores to use for parallel computation. Default is 1.
`x`	The column name for the independent variable. Default is "dose".
`y`	The column name for the dependent variable. Default is "expr".
`other_variables_id_col`	Additional column names for other variables to include in the ANOVA.

Value

anova_res_list A list containing the ANOVA results dataframe, filtered data dictionary, unfiltered ANOVA results dataframe, and a plot list.

`perform_differential_expression_analysis_filtering` *Perform Differential Expression Analysis*

Description

This function filter the data based on their differential analysis

Usage

```
perform_differential_expression_analysis_filtering(
  data_dictionary,
  experimental_data,
  phTable,
  time_point_variable,
  dose_variable,
  samples_variable,
  fcAdjustment = "Nominal",
  fcPval.th = 0.05,
  fc.th = 1.5,
  nCores = 1,
  x = "dose",
  y = "expr",
  other_variables_id_col = NULL
)
```

Arguments

`data_dictionary`	A list of data frames representing the data dictionary.
`experimental_data`	A list of data frames containing the experimental data.
`phTable`	A list of data frames representing the phenotype table.
`time_point_variable`	The name of the variable representing time points.
`dose_variable`	The name of the variable representing doses.
`samples_variable`	The name of the variable representing samples.
`fcAdjustment`	The type of fold change adjustment (default: "Nominal"). Options are "none" or "whatever_other_adjustment".
`fcPval.th`	The p-value threshold for fold change filtering (default: 0.05).
`fc.th`	The fold change threshold (default: 1.5).
`nCores`	The number of cores for parallel processing (default: 1).
`x`	The name of the x-axis variable for plotting (default: "dose").
`y`	The name of the y-axis variable for plotting (default: "expr").
`other_variables_id_col`	A vector of other variable names used for filtering.

Value

A list containing the fold change results (filtered and unfiltered), the filtered data dictionary, and the filtered fold change results.

`perform_trend_test` *Perform the trend test on multiple datasets.*

Description

This function performs the trend test on a list of datasets stored in the `data_dictionary`. The trend test is performed using the `mk.test` function from the `trend` package. It calculates the trend p-value for each dataset and adjusts the p-values using the specified `trendAdjustment` method. It also performs filtering based on the `trendPval.th` threshold.

Usage

```
perform_trend_test(
  data_dictionary,
  trendAdjustment = "Nominal",
  trendPval.th = 0.05,
  trendCores = 1,
  x = "dose",
  y = "expr",
  other_variables_id_col = NULL
)
```

Arguments

`data_dictionary`	A list of datasets, each stored as a data frame in the list.
`trendAdjustment`	The method for adjusting p-values. Default is "Nominal". Available options include "Bonferroni", "Holm", "Hochberg", "BH", "BY", "fdr", "none".
`trendPval.th`	The threshold for filtering p-values. Default is 0.05.
`trendCores`	The number of CPU cores to use for parallel processing. Default is 1.
`x`	The column name representing the predictor variable (independent variable) in each dataset.
`y`	The column name representing the response variable (dependent variable) in each dataset.
`other_variables_id_col`	The column name containing the identifier for other variables. Default is NULL.

Value

A list containing the trend test results and filtered datasets.

plot_bmdx *Plot BMD (Benchmark Dose) Model*

Description

This function generates a plot for a Benchmark Dose (BMD) model.

Usage

```
plot_bmdx(
  model,
  cex = 6,
  confidence_interval = 0.95,
  plot_ic50 = FALSE,
  xlim_u = NULL,
  title_label = "title"
)
```

Arguments

model	The fitted BMD model.
cex	The size of points in the plot (default: 6).
confidence_interval	The confidence interval level (default: 0.95).
title_label	The title of the plot (default: "title").
x_pos	The position of the x-axis (default: 15).
y_pos_th	The position of the y-axis threshold (default: 0.85).

Value

A plot displaying the BMD model with confidence intervals and other related data points.

plot_bmd_bmdl_bmdu_set_of_genes *Plot BMD, BMDL, and BMDU for a Set of Genes*

Description

This function generates a plot showing BMD, BMDL, and BMDU values for a given set of genes.

Usage

```
plot_bmd_bmdl_bmdu_set_of_genes(
  BMDFilMat,
  gi,
  enrich_ppi_info = TRUE,
  gene_id_type = "ENSEMBL",
  organism = "human"
)
```

Arguments

BMDFilMat	A data frame containing BMD values for multiple genes.
gi	A vector of gene names for which to plot BMD values.

Value

A plot showing BMD, BMDL, and BMDU values for the selected set of genes.

---

plot_BMD_tPOD_density *Plot BMD Density with Optional tPOD Reference Lines*

Description

This function generates a density plot of Benchmark Dose (BMD) values and overlays vertical dashed lines for any provided (non-NA) tPOD metrics. The supported metrics include lowest, percentile, first_mode, accumulation, and mean_value. A customizable x-axis range can also be specified.

Usage

```
plot_BMD_tPOD_density(
  BMD_values,
  lowest = NA,
  percentile = NA,
  mean_value = NA,
  accumulation = NA,
  first_mode = NA,
  xrange = NULL,
  title_label = "Density Distribution of BMD Values"
)
```

Arguments

BMD_values	A numeric vector of BMD values.
lowest	Numeric value for the "Lowest" tPOD (optional; default is NA).
percentile	Numeric value for the "Percentile" tPOD (optional; default is NA).
mean_value	Numeric value for the "Mean" tPOD (optional; default is NA).
accumulation	Numeric value for the "Accumulation" tPOD (optional; default is NA).
first_mode	Numeric value for the "First Mode" tPOD (optional; default is NA).
xrange	A numeric vector of length 2 specifying the x-axis range (example is c(0, 5)) for zooming. default is NULL.

Value

A ggplot2 object showing the BMD density distribution and optional vertical tPOD reference lines.

plot_filtering_pie_chart *Plot a pie chart of filtering results*

Description

This function generates a pie chart to visualize the filtering results. The pie chart shows the distribution of variables and non-variables based on the filtering p-values in the table. The function takes the table and an p-value threshold as input.

Usage

```
plot_filtering_pie_chart(tab_unfiltered, Pval.th, title = "")
```

Arguments

tab_unfiltered	The unfiltered table containing the filtering p-values.
Pval.th	The p-value threshold for determining variable significance.
title	Title of the plot. It is a string. default = ""

Value

A pie chart visualization of the filtering results.

plot_gene_pairs *Plot gene pairs comparison*

Description

This function plots the comparison of two gene pairs given the models mod_i and mod_j and newdata containing dose information for new observations.

Usage

```
plot_gene_pairs(mod_i, mod_j, newdata, main = "", feat1 = "g1", feat2 = "g2")
```

Arguments

mod_i	The model for gene pair 1.
mod_j	The model for gene pair 2.
newdata	A data frame containing dose information for new observations.
main	A character string specifying the main title of the plot (default is an empty string).
feat1	A character string specifying the label for gene pair 1 in the plot (default is "g1").
feat2	A character string specifying the label for gene pair 2 in the plot (default is "g2").

Value

A ggplot object showing the comparison of gene pairs.

`plot_histogram` *Plot Histogram and Density*

Description

Generate a histogram and density plot visualization based on provided data and variables.

Usage

```
plot_histogram(
  mod_stats,
  y_val = "BMD",
  color_by = NULL,
  group_by = NULL,
  group_by2 = NULL,
  filter_column = NULL,
  filter_by = NULL,
  alpha_th = 0.5
)
```

Arguments

`mod_stats`	A data frame containing the data for plotting.
`y_val`	The variable to be plotted on the y-axis (numeric).
`color_by`	The categorical variable used for color-coding bars.
`group_by`	The grouping variable for additional data segmentation.
`group_by2`	The second grouping variable for further segmentation.
`filter_column`	The column name for filtering the data.
`filter_by`	The values used for filtering the data.
`alpha_th`	The transparency level of the density plot (ranges between 0-1).

Value

A ggplot2 histogram and density plot visualization.

`plot_pie_chart` *this function allows to plot the histogram of one numeric column of the model statistics*

Description

this function allows to plot the histogram of one numeric column of the model statistics

Usage

```
plot_pie_chart(
  mod_stats,
  category = "Model",
  group_by = NULL,
  group_by2 = NULL,
  filter_column = NULL,
  filter_by = NULL
)
```

Arguments

mod_stats	A data frame containing the data for plotting.
category	The category variable to be used for the pie chart sectors.
group_by	The grouping variable for additional data segmentation.
group_by2	The second grouping variable for further segmentation.
filter_column	The column name for filtering the data.
filter_by	The values used for filtering the data.

Value

a ggplot object

---

plot_scatter *Plot Scatter Plot*

Description

Generate a scatter plot visualization based on provided data and variables.

Usage

```
plot_scatter(
  mod_stats,
  x_val = "BMDL",
  y_val = "BMD",
  color_by = "Model",
  group_by = NULL,
  group_by2 = NULL,
  filter_column = NULL,
  filter_by = NULL
)
```

Arguments

mod_stats	A data frame containing the data for plotting.
x_val	The variable to be plotted on the x-axis.
y_val	The variable to be plotted on the y-axis.
color_by	The categorical variable used for color-coding points.

group_by	The grouping variable for additional data segmentation.
group_by2	The second grouping variable for further segmentation.
filter_column	The column name for filtering the data.
filter_by	The values used for filtering the data.

Value

A ggplot2 scatter plot visualization.

---

point_of_departure *Given a model, this function estimates the Benchmark Dose (BMD), Benchmark Dose Lower Confidence Limit (BMDL), Benchmark Dose Upper Confidence Limit (BMDU), and AC50 values.*

Description

Given a model, this function estimates the Benchmark Dose (BMD), Benchmark Dose Lower Confidence Limit (BMDL), Benchmark Dose Upper Confidence Limit (BMDU), and AC50 values.

Usage

```
point_of_departure(
  model,
  deviation_type = "standard",
  rl = 1.349,
  confidence_interval = 0.95
)
```

Arguments

model	The model object representing the dose-response relationship.
deviation_type	Character string specifying the type of deviation from the fitted model to use for BMD calculation. Default is "standard".
rl	The relative level used to calculate the BMD. Default is 1.349.
confidence_interval	The confidence level for the confidence interval. Default is 0.95.

Value

A modified model object with additional attributes for BMDL, BMDU, and AC50. If an error occurs during the estimation, NA values are assigned to BMDL and BMDU.

pval_adjust *Adjust p-values in filtering results*

Description

This function adjusts the p-values in the filtering results using a specified adjustment method.

Usage

```
pval_adjust(filtering_res, adjustment_method = "fdr")
```

Arguments

filtering_res	The filtering results table.
adjustment_method	The adjustment method for p-values. Options include "Nominal" (no adjustment) and methods supported by the p.adjust function. Default is "fdr" (false discovery rate).

Value

The filtering results table with adjusted p-values and the column "usedFilteringPval" indicating the p-values used for filtering.

read_excel_allsheets *Read all sheets from an Excel file.*

Description

This function reads all sheets from an Excel file specified by the filename. It returns the data as a list of data frames, one for each sheet.

Usage

```
read_excel_allsheets(
  filename,
  tibble = FALSE,
  first_col_as_rownames = FALSE,
  is_rnaseq_raw_count = FALSE,
  check_numeric = TRUE,
  na = "NA"
)
```

Arguments

filename	The path to the Excel file.
tibble	If TRUE, the output data frames will be converted to tibbles. Default is FALSE.
first_col_as_rownames	If TRUE, the first column of each sheet will be used as row names. Default is FALSE.
is_rnaseq_raw_count	If TRUE, the data is assumed to be RNA-Seq raw counts, and it will be converted to log2 counts using the limma::voom function. Default is FALSE.

Value

A list of data frames, one for each sheet in the Excel file.

---

scale_numbers *Scale Numbers to the Range 0-1 This function scales numeric values to the range 0-1 by dividing each value by the maximum value in the input vector.*

Description

Scale Numbers to the Range 0-1 This function scales numeric values to the range 0-1 by dividing each value by the maximum value in the input vector.

Usage

```
scale_numbers(x)
```

Arguments

x	A numeric vector to be scaled.

Value

A numeric vector with scaled values in the range 0-1.

---

select_optimal_models *Select optimal models from a list of current models*

Description

This function selects the optimal models from a list of current models based on the specified method. The supported methods are "AIC" (Akaike Information Criterion) and "Model average". For the "AIC" method, the model with the lowest AIC value is selected for each dataset. For the "Model average" method, if average models are already being computed, the average model is selected; otherwise, the first model in the list is selected. The function returns a list containing the optimal models and a table of computed model statistics.

Usage

```
select_optimal_models(
  current_models,
  method = "AIC",
  time_col_id,
  optional_col_ids = NULL,
  nCores = 1
)
```

Arguments

current_models	The list of current models.
method	The method for selecting the optimal models. Supported values are "AIC" and "Model average".
time_col_id	The identifier for the time column in the model statistics table.
optional_col_ids	Optional identifiers for additional columns in the model statistics table.
nCores	The number of CPU cores to use for parallel computation.

Value

A list containing the optimal models and the computed model statistics table.

---

sym_human_gene_graph *sym_human_gene_graph*

Description

Graph whose nodes are Human genes (Gene Symbols). Edges represent protein-protein interactions (PPIs).

Usage

```
sym_human_gene_graph
```

Format

An object of class `igraph` of length 18932.

Source

https://doi.org/10.1093/bib/bbaa417

sym_human_gene_graph_mirna *sym_human_gene_graph_mirna*

Description

Graph whose nodes are Human genes (Gene Symbols). Edges represent regulatory interactions between miRNA and genes.

Usage

```
sym_human_gene_graph_mirna
```

Format

An object of class `igraph` of length 10291.

Source

https://doi.org/10.1093/bib/bbaa417

sym_human_gene_graph_tf *sym_human_gene_graph_tf*

Description

Graph whose nodes are Human genes (Gene Symbols). Edges represent regulatory interactions between transcription factor (TF) and genes.

Usage

```
sym_human_gene_graph_tf
```

Format

An object of class `igraph` of length 17402.

Source

https://doi.org/10.1093/bib/bbaa417

sym_mouse_gene_graph *sym_mouse_gene_graph*

Description

Graph whose nodes are Mouse genes (Gene Symbols). Edges represent protein-protein interactions (PPIs).

Usage

```
sym_mouse_gene_graph
```

Format

An object of class igraph of length 19323.

Source

https://doi.org/10.1093/bib/bbaa417

sym_mouse_gene_graph_mirna *sym_mouse_gene_graph_mirna*

Description

Graph whose nodes are Mouse genes (Gene Symbols). Edges represent regulatory interactions between miRNA and genes.

Usage

```
sym_mouse_gene_graph_mirna
```

Format

An object of class igraph of length 0.

Source

https://doi.org/10.1093/bib/bbaa417

sym_mouse_gene_graph_tf *sym_mouse_gene_graph_tf*

Description

Graph whose nodes are Mouse genes (Gene Symbols). Edges represent regulatory interactions between transcription factor (TF) and genes.

Usage

```
sym_mouse_gene_graph_tf
```

Format

An object of class `igraph` of length 5062.

Source

https://doi.org/10.1093/bib/bbaa417

sym_rat_gene_graph *sym_rat_gene_graph*

Description

Graph whose nodes are Rat genes (Gene Symbols). Edges represent protein-protein interactions (PPIs).

Usage

```
sym_rat_gene_graph
```

Format

An object of class `igraph` of length 17134.

Source

https://doi.org/10.1093/bib/bbaa417

sym_rat_gene_graph_mirna *sym_rat_gene_graph_mirna*

Description

Graph whose nodes are Rat genes (Gene Symbols). Edges represent regulatory interactions between miRNA and genes.

Usage

```
sym_rat_gene_graph_mirna
```

Format

An object of class `igraph` of length 24.

Source

https://doi.org/10.1093/bib/bbaa417

sym_rat_gene_graph_tf *sym_rat_gene_graph_tf*

Description

Graph whose nodes are Rat genes (Gene Symbols). Edges represent regulatory interactions between transcription factor (TF) and genes.

Usage

```
sym_rat_gene_graph_tf
```

Format

An object of class `igraph` of length 27.

Source

https://doi.org/10.1093/bib/bbaa417

`tpod_accumulation` *Calculate tPOD Using an Accumulation-Based Approach*

Description

This function calculates the tPOD (Point of Departure) by:

1. Identifying the mode and antimode of the input distribution (via `get_mode_antimode`).
2. Filtering the data to values below the antimode.
3. Fitting a shape-constrained additive model (via `scam`).
4. Applying the Kneedle algorithm to the smoothed cumulative sum to detect the knee point.

Usage

```
tpod_accumulation(pod_vector, plot = FALSE, bw_adjust = 3)
```

Arguments

`pod_vector`	A numeric vector representing the POD (Point of Departure) values.
`plot`	Logical. If `TRUE`, a plot is generated showing the sorted data, the fitted curve, a vertical line at the antimode, and another vertical line at the estimated tPOD.
`...`	Additional arguments passed to `get_mode_antimode`.

Details

1. **Sorting**: The function sorts `pod_vector` in increasing order.
2. **Accumulation**: It computes the cumulative sum of the sorted values (y) and the log-transformed x-axis (log10 of the sorted values).
3. **Filtering**: Using the estimated `antimode` from `get_mode_antimode`, data points above the antimode are removed.
4. **Fitting**: A shape-constrained additive model (`scam`) is then fitted to the filtered data to smooth the cumulative sums.
5. **Knee Detection**: The Kneedle algorithm (`kneedle`) is applied to the fitted curve to detect the knee, which is reported as the tPOD.
6. **Optional Plot**: If `plot = TRUE`, the sorted data, fitted curve, and vertical lines marking the antimode and tPOD will be displayed.

Value

A single numeric value representing the tPOD, identified by the Kneedle method on the fitted cumulative distribution.

`tpod_computation` *Compute tPOD using various methods and plot the result*

Description

This function computes the tPOD (Point of Departure) using different methods such as percentile, mean, first mode, lowest, or accumulation After calculating the tPOD, it generates a plot displaying the resulting tPOD value on the accumulation POD distribution.

Usage

```
tpod_computation(
  model_stats,
  tpod_method,
  percentile,
  pod_value = "BMD",
  lowest_method = "lowest",
  bw_adjust = 3,
  ratio_threshold = 1.66,
  plot = FALSE,
  robust = F,
  rank = 1
)
```

Arguments

`model_stats`	A data frame or list containing the model statistics, including the POD values (e.g., BMD, BMDL, or BMDU) from which tPOD will be computed.
`tpod_method`	A string indicating the method to compute tPOD. Options include: "percentile", "mean", "first_mode", "lowest", or "accumulation".
`percentile`	A numeric value between 0 and 1 representing the desired percentile (used if `tpod_method` is "percentile").
`pod_value`	A string indicating which POD value to use from the model statistics. Possible values are "BMD", "BMDL", and "BMDU" (default is "BMD").

Details

The function computes the tPOD using one of the following methods:

- **percentile**: Computes tPOD at the specified percentile of the POD values.
- **mean**: Computes the tPOD as the mean value of the POD values.
- **first_mode**: Computes the tPOD using the first mode (most frequent value) of the POD distribution.
- **lowest**: Computes the tPOD as the lowest value of the POD distribution.
- **accumulation**: Computes the tPOD using the accumulation method.

A plot is also generated showing the accumulation plot of the POD distribution with a vertical line indicating the tPOD value.

Value

A numeric value representing the computed tPOD, according to the selected method.

---

tpod_first_mode *Calculate the first mode value using kernel density estimation*

Description

This function computes the mode of a given vector using kernel density estimation (KDE). The mode is the value that has the highest density in the distribution of the data.

Usage

```
tpod_first_mode(pod_vector)
```

Arguments

pod_vector	A numeric vector for which the mode is to be calculated.

Details

The function uses the kernel density estimation technique to estimate the probability density function of the given vector. It then identifies the value that corresponds to the highest density, which is returned as the mode. This approach is useful when the distribution is not unimodal or not normally distributed.

Value

A numeric value representing the mode of the input vector.

---

tpod_lowest *Calculate the lowest value of a numeric vector, excluding NA values*

Description

This function computes the lowest value in the given numeric vector, excluding any NA values.

Usage

```
tpod_lowest(pod_vector, lowest_method, ratio_threshold, rank)
```

Arguments

pod_vector	A numeric vector for which the lowest value is to be calculated.

Details

The function uses the min function with the na.rm = TRUE argument to exclude NA values when computing the minimum. It returns the lowest value as a numeric result.

Value

A numeric value representing the lowest value of the input vector, excluding NAs.

tpod_mean *Calculate the Mean of a POD Vector*

Description

This function computes the mean of a given POD (Point of Departure) vector, excluding NA values.

Usage

```
tpod_mean(pod_vector)
```

Arguments

pod_vector	A numeric vector containing POD values.

Value

A numeric value representing the mean of the POD vector.

tpod_percentile *Calculate the Percentile of a POD Vector*

Description

This function computes a specified percentile of a given POD (Point of Departure) vector.

Usage

```
tpod_percentile(pod_vector, percentile)
```

Arguments

pod_vector	A numeric vector containing POD values.
percentile	A numeric value representing the desired percentile (should be within 0,1).

Value

A numeric value representing the calculated percentile of the POD vector.

tpod_plot *Generate an accumulation plot with multiple tPOD methods*

Description

This function generates a cumulative sum plot of the given POD vector, and includes vertical lines indicating the tPODs (points of Departure) for multiple methods. The plot can display multiple tPOD values if they all have the same length. The x-axis can optionally be transformed to a logarithmic scale.

Usage

```
tpod_plot(
  pod_vector,
  tpod_method,
  pod_value,
  tPOD,
  xlog = FALSE,
  subtitle = ""
)
```

Arguments

pod_vector	A numeric vector representing the POD (Point of Departure) values.
tpod_method	A character vector containing the names of the tPOD methods used to calculate the points of Departure.
pod_value	A string that represents the value used to calculate the tPODs, for example, "BMD" or "BMDL" or "BMDU".
tPOD	A numeric vector containing the points of Departure (tPODs) corresponding to each method. The length of this vector must match the length of the tpod_method vector.
xlog	A logical value indicating whether to apply a logarithmic transformation to the x-axis (default is FALSE).

Details

The function generates an accumulation plot of the sorted POD vector and adds vertical lines at the locations of the provided tPOD values. The plot can display multiple tPOD values (with different methods) if the length of tPOD and tpod_method are equal. The x-axis can optionally be displayed in logarithmic scale.

Value

A ggplot object representing the cumulative sum plot with vertical lines for tPODs.

upset_plot *Create an UpSet Plot*

Description

This function generates an UpSet plot for the given data, which displays the intersections of sets of genes across different experiments or conditions.

Usage

```
upset_plot(
  mod_stats,
  rel_variable = "Experiment",
  group_by = NULL,
  other_variables = NULL,
  filter_column = c("Model"),
  filter_by = list(c("linear")),
  nintersects = 3,
  group.by = "degree",
  order.by = "degree",
  text.scale = c(1.5, 1.5, 1.2, 1.2, 1.5, 1.5)
)
```

Arguments

mod_stats	A data frame containing the model statistics and gene information.
rel_variable	The name of the variable representing the experiments or conditions.
group_by	The name of the variable to group the data for generating UpSet plots.
other_variables	Additional variables used for plotting, if any.
filter_column	The name of the column to filter the data.
filter_by	A list of filter values to apply on the specified filter_column.
nintersects	The number of intersections to show in the UpSet plot.
group.by	The variable used to group the intersections (e.g., "degree" or "freq").
order.by	The variable used to order the intersections (e.g., "frequency", or "degree").

Value

An UpSet plot displaying the intersections of sets of genes across different experiments or conditions.

Index