Package ‘doseRider’

October 9, 2024

Type Package

Title DoseRider: A multi-omic approach to studying dose-response relationships at the pathway level using mixed models.

Version 1.0.0

Description DoseRider facilitates intricate dose-response analysis in gene expression studies, employing both Linear Mixed Models (LMMs) and Generalized Additive Mixed Models (GAMMs) with cubic splines. It adeptly handles varied omic data, automatically selecting Gaussian or negative binomial distributions based on data type. The package efficiently discerns significant relationships between gene expression and compound doses, comparing null, linear, and non-linear models to ascertain the best fit. DoseRider offers a suite of visualization tools, including plots for random effects and model comparisons, enriching the understanding of gene-specific responses and model dynamics in dose-response studies.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.6.0)

Imports cluster,
ggplot2,
ggrepel,
mgcv,
SummarizedExperiment,
DESeq2,
dplyr,
edgeR,
reshape,
lme4,
performance,
stats,
reshape2,
drc,
tidyr,
utils,
factoextra,
foreach,

doParallel,
parallel,
progress,
doSNOW,
MASS,
splines,
AICcmodavg,
ComplexHeatmap,
circlize,
stringr,
cowplot,
ggridges,
gammit

RoxygenNote 7.3.2

Suggests testthat,
knitr,
rmarkdown

R topics documented:

`add_best_model_adj_pvalue` *Add Best Model Adjusted P-value to DoseRider Object*

Description

This function adds a column "best_model_adj_pvalue" to a DoseRider object, containing the adjusted p-value corresponding to the best model for each gene set.

Usage

```
add_best_model_adj_pvalue(doseRiderObj)
```

Arguments

`doseRiderObj`	A DoseRider object containing results from dose-response analysis.

Value

A DoseRider object with an added column "best_model_adj_pvalue".

Examples

```
## Not run:
updated_results <- add_best_model_adj_pvalue(doseRiderObj)

## End(Not run)
```

`add_cluster_trends_and_bmd` *Add cluster trends and BMD lines to the plot*

Description

Add cluster trends and BMD lines to the plot

Usage

```
add_cluster_trends_and_bmd(
  p,
  gene_set_results,
  mean_data,
  dose_col,
  draw_bmd,
  v_size,
  clusterResults,
  amount
)
```

`add_gene_annotations` *Add gene annotations to the plot*

Description

Add gene annotations to the plot

Usage

```
add_gene_annotations(p, mean_data, dose_col, annotation_text_size)
```

`as.data.frame.DoseRider` *Convert a DoseRider Object to a Data Frame*

Description

This function converts a DoseRider object into a data frame for easier analysis and visualization. It extracts various attributes associated with dose-response analysis results, handling nested list structures.

Usage

```
as.data.frame.DoseRider(object)
```

Arguments

`object`	A DoseRider object containing results from dose-response analysis.

Value

A data frame with attributes from the DoseRider object, where each row corresponds to one gene set.

Examples

```
## Not run: 
  # Assuming `dose_rider_result` is a DoseRider object
  result_df <- as.data.frame.DoseRider(dose_rider_result)

## End(Not run)
```

`bpaf_data` *Transcriptomic Data from Study PRJNA869442: BPA Alternatives in Cultured Breast Cancer Cells*

Description

Overview: Study PRJNA869442 is a transcriptomics study that aims to evaluate potential hazards and compare potencies of Bisphenol A (BPA) and 15 BPA alternative chemicals in cultured breast cancer cells (MCF-7). The study uses high-throughput transcriptomics to examine general toxicological effects and estrogen receptor alpha (ER\alpha)-associated transcriptional changes in response to chemical exposures.

Usage

```
bpaf_data
```

Format

A SummarizedExperiment object or a matrix/data frame with metadata.

Details

Study Design: - Cell Line: MCF-7 breast cancer cells - Exposure: Cells were exposed to BPA and 15 BPA alternative chemicals at concentrations ranging from 0.0005 to 100 µM. - Duration: Exposure period was 48 hours. - Controls: The study includes technical controls (reference RNA, media only, cells with no treatment) and solvent controls (0.1 - Reference Chemicals: The study also includes two reference chemicals, estradiol, and dexamethasone.

Data Format: - Data Type: Gene expression data (RNA-Seq) - Data Format: The data is provided as a SummarizedExperiment object or a matrix/data frame with metadata. - Gene Sets: The data includes information about gene sets and their corresponding expression levels in response to different doses of BPA and its alternative chemicals.

Preprocessing: Before conducting the doseRider analysis, the transcriptomic data from PRJNA869442 has been preprocessed locally to ensure the data's quality and readiness for analysis.

Objective: The objective of this analysis is to utilize doseRider, a function in the doseRider package, to investigate potential non-linear dose-response relationships in the transcriptomic data. By using Generalized Additive Mixed Models (GAMMs), doseRider allows us to understand the relationship between dose-response and gene expression, providing insights into the biological effects and potencies of BPA and its alternative chemicals in cultured breast cancer cells.

Data Source: The original data for study PRJNA869442 can be accessed from the NCBI Sequence Read Archive (SRA). However, for this analysis, we have already preprocessed the data locally and will be using the preprocessed dataset.

Source

The original data for study PRJNA869442 can be accessed from the NCBI Sequence Read Archive (SRA).

See Also

doseRider

Examples

```
# Load the preprocessed dataset
data("bpaf_data")

# Check the structure of the dataset
str(bpaf_data)

# Perform analysis using doseRider
result <- doseRider(bpaf_data)
```

compare_all_models *Compare Models Individually Against the Null Model with ANOVA*

Description

This function compares the goodness of fit of individual models (linear, non-linear fixed effects, and non-linear mixed effects) against a null/base model using ANOVA. P-values are calculated for each model comparison individually.

Usage

```
compare_all_models(
  null_results,
  linear_results = NULL,
  non_linear_fixed_results = NULL,
  non_linear_mixed_results = NULL,
  modelType
)
```

Arguments

`null_results`	A fitted model representing the null or base model.
`linear_results`	A fitted model representing the linear model (optional).
`non_linear_fixed_results`	A fitted model representing the non-linear fixed effects model (optional).
`non_linear_mixed_results`	A fitted model representing the non-linear mixed effects model (optional).
`modelType`	A string indicating the type of model comparison: "LMM" for linear mixed models or "GAMM" for generalized additive mixed models.

Value

A named list of p-values for the comparisons of each model against the base model (null_results). If a model is not provided, its p-value is returned as NA.

Examples

```
# Assuming lmm_null, lmm_linear, lmm_non_linear_fixed, and lmm_non_linear_mixed
# are fitted models:
# compare_models_pvalues <- compare_all_models(lmm_null, lmm_linear, lmm_non_linear_fixed,
# lmm_non_linear_mixed, modelType = "LMM")
```

compute_bmd_bounds *Compute BMD Bounds using Bootstrapping*

Description

This function calculates the lower and upper bounds of the Benchmark Dose (BMD) using bootstrapping on the raw data for significant gene sets.

Usage

```
compute_bmd_bounds(
  dose_rider_results,
  dose_col = "dose",
  sample_col = "sample",
  covariates = c(),
  omic = "rnaseq",
  n_bootstrap = 100,
  ci_level = 0.95,
  filter_type = "fdr",
  threshold = 0.1,
  top = 10,
  model_type = "all"
)
```

Arguments

`dose_rider_results`	A DoseRider object containing results of the analysis.
`dose_col`	Name of the column representing dose information.
`sample_col`	Name of the column representing sample information.
`covariates`	Optional, vector specifying the covariate column(s) in ‘se‘.
`omic`	Type of omics data, defaults to "rnaseq".
`n_bootstrap`	The number of bootstrap samples to generate. Default is 1000.
`ci_level`	Confidence interval level, default is 0.95.
`filter_type`	The type of p-value to filter by. Options are "pvalue" for raw p-value and "fdr" for adjusted p-value.
`threshold`	The p-value threshold for filtering. Defaults to 0.05.
`model_type`	The type of model to filter by. Options are "all", "linear", "non_linear" (which includes "non_linear_fixed" and "non_linear_mixed").

Value

A list containing the lower and upper BMD bounds for each significant gene set.

Examples

```
## Not run:
data("SummarizedExperiment")
gmt <- list(geneSet1 = list(genes = c("gene1", "gene2")))
results <- DoseRider(se, gmt, "dose", "sample", "covariate", "rnaseq", modelType = "GAMM")
bmd_bounds <- compute_bmd_bounds(results, "dose", "sample", n_bootstrap = 1000, ci_level = 0.95)
print(bmd_bounds)

## End(Not run)
```

---

compute_bmd_bounds_parallel

Compute BMD Bounds using Bootstrapping with Parallel Gene Set Processing

---

Description

This function calculates the lower and upper bounds of the Benchmark Dose (BMD) using bootstrapping on the raw data for significant gene sets, with the gene set processing parallelized across multiple cores.

Usage

```
compute_bmd_bounds_parallel(
  dose_rider_results,
  dose_col = "dose",
  sample_col = "sample",
  ci_level = 0.95,
  covariates = c(),
  omic = "rnaseq",
  n_bootstrap = 1000,
  num_cores = 5,
  clusterResults = F
)
```

Arguments

`dose_rider_results`	A DoseRider object containing results of the analysis.
`dose_col`	Name of the column representing dose information.
`sample_col`	Name of the column representing sample information.
`ci_level`	Confidence interval level, default is 0.95.
`covariates`	Optional, vector specifying the covariate column(s) in ‘se‘.
`omic`	Type of omics data, defaults to "rnaseq".
`n_bootstrap`	The number of bootstrap samples to generate. Default is 1000.
`num_cores`	The number of cores to use for parallel processing of gene sets. Default is 5.

Value

A data frame containing the lower and upper BMD bounds for each significant gene set.

Examples

```
## Not run:
data("SummarizedExperiment")
gmt <- list(geneSet1 = list(genes = c("gene1", "gene2")))
results <- DoseRider(se, gmt, "dose", "sample", "covariate", "rnaseq", modelType = "GAMM")
bmd_bounds <- compute_bmd_bounds_parallel(results, "dose", "sample", n_bootstrap = 1000, ci_level = 0.95, num_core
print(bmd_bounds)

## End(Not run)
```

---

compute_bmd_for_gene_in_geneset

Compute Benchmark Dose (BMD) for a Specific Gene within a Gene Set

---

Description

This function calculates the Benchmark Dose (BMD) for a specific gene within a given gene set based on smoothed trend data from DoseRider analysis. The BMD is identified as the dose level where the predicted response for the specific gene exceeds a threshold defined as a specified number of standard deviations (‘z‘) above the control response.

Usage

```
compute_bmd_for_gene_in_geneset(
  dose_rider_results,
  gene_set_name,
  gene_name,
  dose_var = "Dose",
  z = 1
)
```

Arguments

`dose_rider_results`	A list containing the results of DoseRider analysis.
`gene_set_name`	The name of the gene set.
`gene_name`	The name of the specific gene within the gene set.
`dose_var`	The name of the dose variable in the results. Default is "Dose".
`z`	A numeric value specifying the number of standard deviations above the control response to define the target response for BMD. Default is 1.

Value

A numeric value representing the calculated BMD for the specified gene. If no BMD is found, NA is returned.

Examples

```
## Not run:
dose_rider_results <- DoseRider(se, gmt, "dose", "sample")
bmd <- compute_bmd_for_gene_in_geneset(dose_rider_results, "geneSet1", "gene1")
print(bmd)

## End(Not run)
```

`compute_bmd_from_main_trend`
Compute Benchmark Dose (BMD) from Smoothed Trend

Description

This function calculates the Benchmark Dose (BMD) based on a smoothed trend from model predictions. The BMD is identified as the dose level where the predicted response exceeds a threshold defined as a specified number of standard deviations (z) above the control response.

Usage

```
compute_bmd_from_main_trend(
  smooth_pathway,
  dose_var,
  z = 1,
  center_values = TRUE,
  scale_values = F
)
```

Arguments

`smooth_pathway`	Data frame containing smoothed trend predictions, typically from a dose-response model.
`dose_var`	The name of the dose variable in smooth_pathway.
`z`	A numeric value specifying the number of standard deviations above the control response to define the target response for BMD. Default is 1.
`center_values`	Logical, if TRUE the predictions are centered around their mean. Default is TRUE.

Value

A numeric value representing the calculated BMD. If no BMD is found within a reasonable range, NA is returned.

Examples

```
## Not run:
smooth_pathway <- data.frame(Dose = c(0, 1, 2, 3), predictions = c(0.1, 0.5, 0.9, 1.2))
bmd <- compute_bmd_from_main_trend(smooth_pathway, "Dose")
print(bmd)

## End(Not run)
```

compute_derivatives *Compute Derivatives and Identify Zero Points*

Description

This function calculates the first and second derivatives of the predicted values from a mixed model and identifies points where these derivatives are approximately zero.

Usage

```
compute_derivatives(smooth_pathway, dose_var, tolerance = 0.001)
```

Arguments

smooth_pathway	Data frame containing smoothed trend predictions, typically from a dose-response model.
dose_var	The name of the dose variable in ‘smooth_pathway‘.

Value

A list containing the first derivative, second derivative, and the zero points for both first and second derivatives.

Examples

```
## Not run:
smooth_pathway <- data.frame(Dose = c(0, 1, 2, 3), predictions = c(0.1, 0.5, 0.9, 1.2))
derivatives <- compute_derivatives(smooth_pathway, "Dose")
print(derivatives)

## End(Not run)
```

`compute_IC50` *Compute IC50 from SummarizedExperiment object*

Description

This function computes the IC50 (half-maximal inhibitory concentration) for each gene in a given ‘SummarizedExperiment‘ object based on the dose-response relationship. The function takes the name of the dose column and the viability column (response) from the ‘colData‘ and returns a vector of IC50 values.

Usage

```
compute_IC50(se, dose_col, viability_col)
```

Arguments

`se`	A ‘SummarizedExperiment‘ object containing the dose-response data.
`dose_col`	A string representing the name of the column in ‘colData‘ that contains the dose information.
`viability_col`	A string representing the name of the column in ‘colData‘ that contains the viability or response data.

Value

A named vector of IC50 values for each gene, where the names are the gene identifiers.

Examples

```
# Assuming `se` is a SummarizedExperiment object with dose and viability information
# ic50_values <- compute_IC50(se, dose_col = "dose", viability_col = "viability")
```

`compute_metrics_gamm` *Compute metrics for a given gam or bam model*

Description

This function computes the AIC, BIC, and effective degrees of freedom (edf) for a given ‘gam‘ or ‘bam‘ model.

Usage

```
compute_metrics_gamm(model)
```

Arguments

`model`	A ‘gam‘ or ‘bam‘ model object from the mgcv package.

Details

The approach to calculate the effective degrees of freedom (edf) is adapted from the itsadug package: Interpreting Time Series and Autocorrelated Data Using GAMMs Reference: itsadug package (https://CRAN.R-project.org/package=itsadug)

Value

A list containing the AIC, BIC, and edf of the model. If the model parameter isn’t a ‘gam‘ or ‘bam‘ object, the function will return NA for all metrics.

Examples

```
library(mgcv)
data(sleepstudy)
gam_model <- gam(Reaction ~ s(Days), data = sleepstudy)

# Compute metrics
compute_metrics_gamm(gam_model)
```

`compute_metrics_lmm` *Compute metrics for a given lmer or glmer model*

Description

This function computes the AIC, AICc, BIC, and degrees of freedom metrics for a given ‘lmer‘ or ‘glmer‘ model.

Usage

```
compute_metrics_lmm(model)
```

Arguments

`model`	A ‘lmer‘ or ‘glmer‘ model object. If it’s neither, the function returns NA for all the metrics.

Value

A list containing the AIC, BIC, and degrees of freedom of the model. If the model parameter isn’t a ‘lmer‘ or ‘glmer‘ object, the function will return NA for all metrics.

Examples

```
library(lme4)
data(sleepstudy)
model <- lmer(Reaction ~ Days + (1|Subject), data = sleepstudy)

# Compute metrics
compute_metrics(model)
```

---

create_gamm_formula — *Create Generalized Additive Mixed Model (GAMM) Formula with Omics Consideration*

Description

Generates a formula for fitting a GAMM, given response, fixed effects, random effects, omics type, and model type. Supports 'base', 'linear', and 'cubic' models and can handle RNA-seq data.

Usage

```
create_gamm_formula(
  response,
  fixed_effects,
  random_effects,
  covariates = c(),
  model_type = "non_linear",
  omic = NULL,
  k = NULL
)
```

Arguments

`response`	A string specifying the response variable in the model.
`fixed_effects`	A string or a vector of strings specifying the fixed effects in the model.
`random_effects`	A string or a vector of strings specifying the random effects in the model.
`covariates`	Optional vector of additional covariates in the model.
`model_type`	Type of the model ('base', 'linear', 'cubic').
`omic`	Type of omic data ('base' or 'rnaseq').

Value

A string representing the GAMM formula.

Examples

```
## Not run:
 base_formula <- create_gamm_formula("mpg", "hp", "cyl", model_type = "base", omic = "base")
 cubic_formula <- create_gamm_formula("mpg", "hp", "cyl", model_type = "cubic", omic = "rnaseq")

## End(Not run)
```

`create_gene_heatmap` *Create Heatmap for Individual Genes within a Gene Set*

Description

This function creates a heatmap for a specified gene set from DoseRider results. Each row in the heatmap represents an individual gene, and each column represents different doses, showing gene expression values.

Usage

```
create_gene_heatmap(
  dose_rider_results,
  gene_set_name,
  dose_col,
  dose_unit = "M",
  fontsize = 6
)
```

Arguments

`dose_rider_results`	A list containing the results of the DoseRider analysis for each gene set.
`gene_set_name`	A character string specifying the name of the gene set to be visualized.
`dose_col`	A character string specifying the name of the dose column in the raw expression data.
`dose_unit`	A character string specifying the dose units to plot in the column title..
`fontsize`	Integer for fontsize. Default 6

Value

An object of class ‘Heatmap‘ representing the constructed heatmap for the specified gene set.

Examples

```
## Not run:
  # Assuming `dose_rider_results` is your DoseRider analysis result
  heatmap_plot <- create_gene_heatmap(dose_rider_results, "Gene Set Name", "dose")
  draw(heatmap_plot) # To display the heatmap

## End(Not run)
```

`create_lmm_formula` *Create Generalized Linear Mixed Model (GLMM) Formula*

Description

This function generates a formula for a GAMM, given response and effect variables. The main goal is to compare the 'cubic' model with the 'base' model.

Usage

```
create_lmm_formula(
  response,
  fixed_effects,
  random_effects,
  covariates = c(),
  model_type = "base",
  omic = NULL,
  spline_knots = 3
)
```

Arguments

`response`	A string specifying the response variable in the model.
`fixed_effects`	A string or a vector of strings specifying the fixed effects in the model.
`random_effects`	A string or a vector of strings specifying the random effects in the model.
`covariates`	A string or a vector of strings specifying any covariates to be included in the model. Default is an empty vector.
`model_type`	A string specifying the type of the model. Can be 'base' or 'cubic'. Default is 'base'.
`omic`	A character string specifying the type of data. If 'rnaseq', an offset is included in the formula.

Value

A string representing the formula for the GAMM.

Examples

```
## Not run:
# Create a base formula
base_formula <- create_gamm_formula(response = "counts",
                                    fixed_effects = "dose",
                                    random_effects = "gene",
                                    model_type = "base",
                                    omic = "rnaseq")
cubic_formula <- create_gamm_formula(response = "counts",
                                       fixed_effects = "dose",
                                       random_effects = "gene",
                                       model_type = "cubic",
                                       omic = "rnaseq")

## End(Not run)
```

---

create_summarized_experiment

Create a SummarizedExperiment object from assay and metadata data frames

Description

This function creates a SummarizedExperiment object which combines the assay data (counts, or other measurements) and associated metadata.

Usage

```
create_summarized_experiment(assay_df, metadata_df)
```

Arguments

`assay_df`	Data frame containing assay data, with rows as genes and columns as samples.
`metadata_df`	Data frame containing metadata for the samples.

Value

A SummarizedExperiment object with the given assay and metadata.

Examples

```
assay_df <- data.frame(matrix(runif(100), nrow=10))
metadata_df <- data.frame(condition = rep(c("A", "B"), each = 5))
se <- create_summarized_experiment(assay_df, metadata_df)
```

DoseRider

Perform DoseRider Analysis Using Linear Mixed Models (LMMs) or Generalized Additive Mixed Models (GAMMs)

Description

This function performs DoseRider analysis on gene expression data, applying either Linear Mixed Models (LMMs) or Generalized Additive Mixed Models (GAMMs) to each gene set defined in the gene matrix transposed (GMT) format. It evaluates dose-response relationships in the context of gene sets and calculates various model metrics, significance, and smoothing predictions.

Usage

```
DoseRider(
  se,
  gmt,
  dose_col = "dose",
  sample_col = "sample",
  covariates = c(),
  omic = "rnaseq",
  minGSsize = 5,
  maxGSsize = 300,
  method = "fdr",
  modelType = "LMM",
  FilterPathway = FALSE,
  pca_threshold = 0.6,
  log_transform = F,
  models = c("linear", "non_linear_fixed", "non_linear_mixed")
)
```

Arguments

`se`	SummarizedExperiment object or a matrix/data frame containing gene expression data.
`gmt`	List of gene sets, each represented as a list with gene names.
`dose_col`	Name of the column representing dose information.
`sample_col`	Name of the column representing sample information.
`covariates`	Optional, vector specifying the covariate column(s) in ‘se‘.
`omic`	Type of omics data, defaults to "rnaseq".
`minGSsize`	Minimum gene set size for analysis, defaults to 5.
`maxGSsize`	Maximum gene set size for analysis, defaults to 300.
`method`	Method for multiple testing adjustment, defaults to "fdr".
`modelType`	Type of model to be used for analysis, "LMM" for Linear Mixed Models or "GAMM" for Generalized Additive Mixed Models. Defaults to "LMM".
`FilterPathway`	Boolean, if TRUE the function will apply PCA filtering to detect antagonist patterns. Defaults to FALSE.
`pca_threshold`	Numeric value specifying the variance threshold for PC1 to filter pathways. Default is 0.6.
`log_transform`	Logical, whether to log10 transform the dose values. Default is FALSE.

Value

A list containing results for each gene set including various metrics, p-values, and adjusted p-values.

Examples

```
## Not run:
data("SummarizedExperiment")
gmt <- list(geneSet1 = list(genes = c("gene1", "gene2")))
results <- DoseRider(se, gmt, "dose", "sample", "covariate", "rnaseq", modelType = "GAMM")

## End(Not run)
```

`DoseRiderParallel` *Perform DoseRider analysis in parallel using multiple cores.*

Description

This function conducts DoseRider analysis in parallel, applying either Linear Mixed Models (LMMs) or Generalized Additive Mixed Models (GAMMs) to each gene set in the gene matrix transposed (GMT) format. It evaluates dose-response relationships in gene sets, computing various model metrics and significance.

Usage

```
DoseRiderParallel(
  se,
  gmt,
  dose_col = "dose",
  sample_col = "sample",
  covariates = c(),
  omic = "rnaseq",
  minGSsize = 5,
  maxGSsize = 300,
  method = "fdr",
  num_cores = 5,
  modelType = "LMM",
  FilterPathway = FALSE,
  pca_threshold = 0.6,
  log_transform = F,
  models = c("linear", "non_linear_fixed", "non_linear_mixed")
)
```

Arguments

`se`	The input SummarizedExperiment object or matrix/data frame with metadata.
`gmt`	The gene set collection as a list with gene sets.
`dose_col`	The name of the column in the metadata representing the dose information.
`sample_col`	The name of the column in the metadata representing the sample information.

`covariates`	The name of the column in the metadata representing the covariate information (optional).
`omic`	The type of omic data used (default is "rnaseq").
`minGSsize`	The minimum gene set size for filtering (default is 5).
`maxGSsize`	The maximum gene set size for filtering (default is 300).
`method`	The p-value adjustment method for FDR correction (default is "fdr").
`num_cores`	The number of cores to use for parallel processing (default is 5).
`modelType`	Type of model to be used for analysis, "LMM" for Linear Mixed Models or "GAMM" for Generalized Additive Mixed Models. Defaults to "LMM".
`FilterPathway`	Boolean, if TRUE the function will apply PCA filtering to detect antagonist patterns. Defaults to FALSE.
`pca_threshold`	Numeric value specifying the variance threshold for PC1 to filter pathways. Default is 0.6.
`log_transform`	Logical, whether to log10 transform the dose values. Default is FALSE.

Value

A list containing the results of the DoseRider analysis for each gene set.

Examples

```
## Not run:
data("SummarizedExperiment")
gmt <- list(geneSet1 = list(genes = c("gene1", "gene2")))
results <- DoseRiderParallel(se, gmt, "dose", "sample", "covariate", "rnaseq", num_cores = 4, modelType = "GAMM")

## End(Not run)
```

`dose_response_heatmap` *Create Complex Heatmap for Gene Sets*

Description

This function creates a heatmap using the ‘ComplexHeatmap‘ package, where each row represents a gene set and each column represents the average expression of genes within that gene set for each dose.

Usage

```
dose_response_heatmap(
  dose_rider_results,
  dose_col = "Dose",
  dose_unit = "M",
  top = 15,
  order_column = "NegLogPValue",
  decreasing = FALSE,
  fontsize = 6
)
```

Arguments

`dose_rider_results`	A list containing the results of the DoseRider analysis for each gene set. Each element of the list is a sublist with various metrics and the raw values (expression data) for a gene set.
`dose_col`	A character string specifying the name of the dose column in the raw expression data.
`dose_unit`	A character string specifying the dose units to plot in the column title..
`top`	An integer specifying the number of top gene sets to include in the heatmap. Default is 15.
`order_column`	A character string specifying the column to use for ordering gene sets in the heatmap.
`fontsize`	Integer for fontsize. Default 6

Value

An object of class 'Heatmap' representing the constructed heatmap.

Examples

```
## Not run:
  # Assuming `dose_rider_results` is your DoseRider analysis result
  heatmap_plot <- create_complex_heatmap(dose_rider_results, "dose")
  draw(heatmap_plot) # To display the heatmap

## End(Not run)
```

`estimate_model_parameters` *Update SummarizedExperiment with Estimated Model Parameters*

Description

Estimates necessary parameters for DESeq2 or edgeR and updates the SummarizedExperiment object with size factors, dispersions, and theta values in its rowData and colData.

Usage

```
estimate_model_parameters(se)
```

Arguments

`se`	SummarizedExperiment object containing count data and sample metadata.

Value

Updated SummarizedExperiment object with size factors, dispersions, and theta values included.

`extract_bmd_for_pathway` *Extract Benchmark Dose (BMD) Values for a Pathway*

Description

This function extracts the Benchmark Dose (BMD) values for all clusters within a specified pathway from the DoseRider results.

Usage

```
extract_bmd_for_pathway(dose_rider_results, gene_set_name)
```

Arguments

`dose_rider_results`	A list containing the results of DoseRider analysis.
`gene_set_name`	The name of the gene set/pathway.

Value

A numeric vector containing the BMD values for the specified pathway. If no cluster-specific results are available, NA is returned.

Examples

```
## Not run:
dose_rider_results <- DoseRider(se, gmt, "dose", "sample")
bmd_values <- extract_bmd_for_pathway(dose_rider_results, "geneSet1")
print(bmd_values)

## End(Not run)
```

extract_gene_set_results *Extract gene set results from dose_rider_results*

Description

Extract gene set results from dose_rider_results

Usage

```
extract_gene_set_results(dose_rider_results, gene_set_name, plot_original_data)
```

extract_random_effects_gamm *Extract Random Effects from GAMM*

Description

This function extracts the random effect estimates for each gene from a GAMM object. The result is a dataframe with genes as rows and their corresponding random effect values.

Usage

```
extract_random_effects_gamm(model, dose_col)
```

Arguments

model	A GAMM model object.

Value

A dataframe with genes as rows and their random effect estimates.

`extract_random_effects_lmm`
Extract Random Effects from LMM

Description

This function extracts the random effect estimates for each gene from an LMM model object. The result is a dataframe with genes as rows and their corresponding random effect values.

Usage

```
extract_random_effects_lmm(model, dose_col)
```

Arguments

`model`	An LMM model object, typically of class 'lmerMod'.

Value

A dataframe with genes as rows and their random effect estimates.

`extract_tcd_for_pathway`
Extract TCD (Compute Derivatives and Identify Zero Points) Values for a Pathway

---

Description

This function extracts the TCD (Threshold Concentration Dose) values for all clusters within a specified pathway from the DoseRider results.

Usage

```
extract_tcd_for_pathway(dose_rider_results, gene_set_name)
```

Arguments

`dose_rider_results`	A list containing the results of DoseRider analysis.
`gene_set_name`	The name of the gene set/pathway.

Value

A numeric vector containing the TCD values for the specified pathway. If no cluster-specific results are available, NA is returned.

Examples

```
## Not run:
dose_rider_results <- DoseRider(se, gmt, "dose", "sample")
tcd_values <- extract_tcd_for_pathway(dose_rider_results, "geneSet1")
print(tcd_values)

## End(Not run)
```

---

filter_DoseRider *Filter DoseRider Object*

Description

This function filters a DoseRider object based on the specified model type and p-value criteria.

Usage

```
filter_DoseRider(
  doseRiderObj,
  model_type = "all",
  filter_type = "pvalue",
  threshold = 0.05
)
```

Arguments

doseRiderObj	A DoseRider object containing results from dose-response analysis.
model_type	The type of model to filter by. Options are "all", "linear", "non_linear" (which includes "non_linear_fixed" and "non_linear_mixed").
filter_type	The type of p-value to filter by. Options are "pvalue" for raw p-value and "fdr" for adjusted p-value.
threshold	The p-value threshold for filtering. Defaults to 0.05.

Value

A filtered DoseRider object.

Examples

```
## Not run:
filtered_results <- filter_DoseRider(doseRiderObj, model_type = "linear", filter_type = "fdr", threshold = 0.05)

## End(Not run)
```

`filter_gmt_by_id` *Filter Gene Sets in a GMT File by Specific IDs*

Description

This function filters gene sets contained within a GMT file based on a given vector of IDs. It checks each gene set for a matching ‘external_id‘ and retains only those that match any of the provided IDs.

Usage

```
filter_gmt_by_id(gmt, vector_id)
```

Arguments

`gmt`	A list representing the GMT file's data. Each element of the list is expected to be a gene set, which is itself a list with an ‘external_id‘ field.
`vector_id`	A vector of IDs (as character strings or numeric values). Gene sets whose ‘external_id‘ matches any of these IDs will be retained in the output.

Value

A list of gene sets from the GMT file that match the specified IDs. This list will only contain elements where the ‘external_id‘ matched at least one of the IDs in ‘vector_id‘. If no matches are found, an empty list is returned.

Note

This function does not handle cases where ‘external_id‘ is missing in the gene sets. It is assumed that each gene set in the GMT has a valid ‘external_id‘ field.

Examples

```
# Assuming `gmt_data` is a list representing a GMT file
# and you want to filter for gene sets with external IDs 101 and 102:
filtered_data <- filter_gmt_by_id(gmt_data, c(101, 102))
```

`filter_gmt_by_size` *Filter GMT by Geneset Size*

Description

This function filters a GMT (Gene Matrix Transposed file format) object by a specified geneset size range. Each geneset in the GMT object is checked to see if its size falls within the minimum and maximum size thresholds. If it does, the geneset is kept; if it doesn't, the geneset is discarded.

Usage

```
filter_gmt_by_size(gmt, minGenesetSize, maxGenesetSize)
```

Arguments

`gmt`	A list, where each element represents a geneset. Each geneset should be a named list with a field for the geneset name and a field for the genes contained in the geneset.
`minGenesetSize`	An integer specifying the minimum number of genes that a geneset must contain to be kept.
`maxGenesetSize`	An integer specifying the maximum number of genes that a geneset can contain to be kept.

Value

A list (like the original GMT object) but only containing the genesets that met the size criteria.

Examples

```
# Consider a GMT object
gmt <- list(list(pathway = "geneset1", genes = c("gene1", "gene2", "gene3")),
             list(pathway = "geneset2", genes = c("gene4", "gene5")),
         list(pathway = "geneset3", genes = c("gene6", "gene7", "gene8", "gene9", "gene10")))

# Filter the GMT object
filter_gmt_by_size(gmt, minGenesetSize = 3, maxGenesetSize = 4)
# This will return the first geneset only, as it is the only one with size between 3 and 4
```

filter_pathway_by_pca *Filter Pathways Based on PCA to Detect Antagonist Patterns*

Description

This function filters pathways in transcriptomic data by performing Principal Component Analysis (PCA). It determines whether to keep or skip a pathway based on the variance explained by the first principal component (PC1), and by checking for antagonist patterns in the loadings of PC1.

Usage

```
filter_pathway_by_pca(
  long_df,
  dose_col = "Dose",
  pca_threshold = 0.7,
  expression_col = "counts",
  loading_threshold = 0.6,
  antagonist_threshold = 2
)
```

Arguments

`long_df`	Dataframe containing the long format transcriptomic data.
`dose_col`	Character string specifying the name of the dose column in ‘long_df‘. Default is "Dose".
`pca_threshold`	Numeric value specifying the variance threshold for PC1 to filter pathways. Default is 0.6. Higher values are more restrictive, requiring PC1 to explain a larger portion of variance.
`expression_col`	Character string specifying the name of the expression column in ‘long_df‘. Default is "counts".
`loading_threshold`	Numeric value specifying the loading threshold to detect antagonist patterns. Default is 0.5. Lower values are more restrictive, identifying more subtle antagonistic patterns.
`antagonist_threshold`	Numeric value specifying the threshold for detecting antagonist patterns. Default is 0.5. Lower values are more restrictive, identifying more subtle antagonistic patterns, while higher values are less restrictive.

Value

Logical value: TRUE if the pathway should be kept (no antagonist pattern detected), FALSE otherwise.

Examples

```
## Not run:
  # Assuming 'long_df' is available and transform_smooth_values function is defined
 should_keep_pathway <- filter_pathway_by_pca(long_df, dose_col = "Dose", pca_threshold = 0.7, expression_col = "
  if (should_keep_pathway) {
    # Proceed with further analysis
  }

## End(Not run)
```

---

find_geneset_index *Find the Index of a Geneset in a GMT Object*

Description

This function searches for a specified geneset within a GMT (Gene Matrix Transposed file format) object and returns its index. If the geneset is not found, NULL is returned.

Usage

```
find_geneset_index(gmt, geneset_name, pathway_col = "pathway")
```

Arguments

`gmt`	A list, where each element represents a geneset. Each geneset should be a named list with at least a field corresponding to the geneset name.
`geneset_name`	A character string representing the name of the geneset to find.
`pathway_col`	A character string representing the name of the field in the geneset list items which corresponds to the geneset name.

Value

The index of the geneset in the GMT object if found, otherwise NULL.

Examples

```
# Consider a GMT object
gmt <- list(list(pathway = "geneset1", gene_ids = c("gene1", "gene2")),
             list(pathway = "geneset2", gene_ids = c("gene3", "gene4")))

# Find the index of a geneset
find_geneset_index(gmt, "geneset2", "pathway") # returns 2
find_geneset_index(gmt, "nonexistent_geneset", "pathway") # returns NULL
```

fit_gam — *Fit Generalized Additive Model (GAM)*

Description

This function fits a GAM to the provided data using the specified formula. It adapts to different omics types and handles family specification for RNAseq data.

Usage

```
fit_gam(formula, data, omic)
```

Arguments

`formula`	A formula for the GAM.
`data`	A data frame containing the data for modeling.
`omic`	A character string specifying the type of omic data. If 'rnaseq', a negative binomial family is used.

Value

A GAM model if fitting is successful; NA otherwise.

Examples

```
## Not run:
  data("mtcars")
  formula <- mpg ~ s(hp) + s(wt)
  model <- fit_gam(formula, data = mtcars, omic = "base")

## End(Not run)
```

fit_lmm — *Fit Linear Mixed Model (LMM) or Generalized Linear Mixed Model (GLMM)*

Description

This function fits an LMM or a GLMM to the provided data using the specified formula. The aim is to model all the genes from a specified pathway to check the effects of a dose. The model's family is chosen based on the 'omic' parameter.

Usage

```
fit_lmm(formula, data, omic = "base")
```

Arguments

`formula`	A formula for the LMM or GLMM.
`data`	A data frame (in long format) containing the data to be modeled. The data frame should include columns for dose, gene (or metabolite), and, if 'rnaseq' is specified, an offset. Other covariates may also be present.
`omic`	A character string specifying the type of data. If 'rnaseq', the family is set to negative binomial.
`theta`	The theta parameter for the negative binomial family. Relevant only if omic = "rnaseq".

Value

An LMM or GLMM model if fitting is successful; NA otherwise.

Examples

```
## Not run:
data("mtcars")
# Generate a suitable formula for the LMM
formula <- mpg ~ hp + (1|cyl)
model <- fit_lmm(formula = formula, data = mtcars, omic = "rnaseq", theta = 1.5)

## End(Not run)
```

`fit_model_compute_bmd` *Fit Model and Compute BMD on Bootstrapped Data*

Description

This function fits a model to a bootstrapped sample and computes the BMD, either for the entire gene set or for clusters within the gene set.

Usage

```
fit_model_compute_bmd(
  long_df,
  formula,
  omic = "rnaseq",
  clusterResults = FALSE,
  dose_col
)
```

Arguments

`long_df`	Data frame containing the raw data for the gene set.
`formula`	The formula used to fit the model (LMM or GAMM).
`omic`	Type of omics data, defaults to "rnaseq".
`clusterResults`	Boolean, if TRUE, compute BMD for clusters; otherwise, for the whole gene set. Default is FALSE.
`dose_col`	Name of the column representing dose information.

Value

A list containing the BMD for the bootstrapped sample.

Examples

```
## Not run:
long_df_bootstrap <- long_df[sample(nrow(long_df), replace = TRUE), ]
bootstrap_results <- fit_model_compute_bmd(long_df_bootstrap, formula, "rnaseq", clusterResults = FALSE, dose_col
print(bootstrap_results)

## End(Not run)
```

---

get_bmd_range *Compute BMD Range and Find Peaks*

Description

This function computes the range of Benchmark Dose (BMD) values from the DoseRider results and identifies peaks in the density of BMD values.

Usage

```
get_bmd_range(dose_rider_results)
```

Arguments

`dose_rider_results`	A list containing the results of DoseRider analysis.

Value

A list containing the BMD values (‘x‘), their density (‘y‘), and the identified peaks (‘bmd‘).

Examples

```
## Not run:
dose_rider_results <- DoseRider(se, gmt, "dose", "sample")
bmd_range <- get_bmd_range(dose_rider_results)
print(bmd_range)

## End(Not run)
```

`get_tcd_range` *Compute TCD Range and Find Zero Points*

Description

This function computes the range of TCD (Threshold Concentration Dose) values from the DoseRider results and identifies zero points in the density of TCD values.

Usage

```
get_tcd_range(dose_rider_results)
```

Arguments

`dose_rider_results`	A list containing the results of DoseRider analysis.

Value

A list containing the TCD values (‘x‘), their density (‘y‘), and the identified zero points (‘tcd‘).

Examples

```
## Not run:
dose_rider_results <- DoseRider(se, gmt, "dose", "sample")
tcd_range <- get_tcd_range(dose_rider_results)
print(tcd_range)

## End(Not run)
```

`get_top_genesets` *Function to obtain the top k gene sets sorted by a specified column*

Description

Function to obtain the top k gene sets sorted by a specified column

Usage

```
get_top_genesets(
  dose_rider_results,
  pvalue_column = "best_model_adj_pvalue",
  order_column = "NegLogPValue",
  top = 10,
  decreasing = TRUE
)
```

Arguments

`dose_rider_results`	Data frame containing gene set information
`pvalue_column`	Column name for p-values (default: "best_model_adj_pvalue")
`order_column`	Column name for ordering (default: "NegLogPValue")
`top`	Number of top gene sets to retrieve (default: 10)
`decreasing`	Logical indicating sort order (default: TRUE)

Value

Vector of gene set names of the top k gene sets

`initialize_plot` *Initialize the ggplot object with gene-specific trends*

Description

Initialize the ggplot object with gene-specific trends

Usage

```
initialize_plot(mean_data, dose_col, model_metrics, gene_set_name)
```

`loadCPDB` *Load CPDB GMT File*

Description

This function loads the GMT file corresponding to the desired identifier from the ConsensusPathDB.

Usage

```
loadCPDB(identifier)
```

Arguments

`identifier`	A character string specifying the desired identifier. Options include 'ENSEMBL', 'Entrez', 'RefSeq', 'Symbol', 'Metabolites'.

Value

The loaded GMT file as a data frame.

Examples

```
loadCPDB("ENSEMBL")
```

`optimal_clusters_silhouette` *Determine Optimal Number of Clusters Using Silhouette Method*

Description

This function determines the optimal number of clusters for dose-response data using the silhouette method. It returns the optimal number of clusters and the cluster assignments for each gene.

Usage

```
optimal_clusters_silhouette(data, dose_col, max_clusters = 10)
```

Arguments

`data`	Dataframe containing dose-response data.
`dose_col`	Character string specifying the name of the dose column in 'data'.
`max_clusters`	Maximum number of clusters to consider. Default is 10.

Value

A list containing the optimal number of clusters and a dataframe of cluster assignments.

Examples

```
## Not run:
  # Assuming 'data' is available
  cluster_results <- optimal_clusters_silhouette(data, dose_col = "Dose")

## End(Not run)
```

`plot_bmd_confidence_intervals`
Plot Bubble Plot of BMD with Confidence Intervals

Description

This function creates a bubble plot visualizing the BMD results with confidence intervals. The size of the bubbles represents the median BMD, while the error bars show the confidence intervals.

Usage

```
plot_bmd_confidence_intervals(bmd_bounds_df, top = 10)
```

Arguments

`bmd_bounds_df`	A data frame containing the BMD results with columns for lower bound, upper bound, mean BMD, and median BMD.

Value

A ggplot object representing the bubble plot of BMD with confidence intervals.

Examples

```
## Not run:
  # Assuming bmd_bounds_df is available
  bmd_plot <- plot_bmd_confidence_intervals(bmd_bounds_df, top = 10)
  print(bmd_plot)

## End(Not run)
```

plot_bmd_density_and_peaks

Plot Benchmark Dose (BMD) Density and Peaks

Description

This function creates a plot visualizing the density of BMD values and highlights the peaks where the highest density of BMD values are found.

Usage

```
plot_bmd_density_and_peaks(bmd_range_output, log_bmd = T)
```

Arguments

`bmd_range_output`	A list containing the output from 'get_bmd_range' function, which includes x (BMD values), y (density), and bmd (peaks).
`log_bmd`	Bool. If true compute the BMD text as 10 exponential

Value

A ggplot object visualizing the density of BMD values with peaks marked.

Examples

```
bmd_range_output <- get_bmd_range(dose_rider_results)
plot_bmd_density_and_peaks(bmd_range_output)
```

plot_dotplot_top_pathways

Plot Dot Plot of Top Pathways from DoseRider Results

Description

This function creates a dot plot visualizing the top pathways from DoseRider analysis results based on their adjusted cubic p-value. The size of the dots represents the number of genes in each pathway, while the color indicates the best model selected for each pathway.

Usage

```
plot_dotplot_top_pathways(
  dose_rider_results,
  top = 10,
  order_column = "NegLogPValue",
  pvalue_column = "best_model_pvalue",
  decreasing = F
)
```

Arguments

`dose_rider_results`	A list containing the results from the DoseRider analysis.
`top`	The number of top pathways to display in the plot. Default is 10.
`order_column`	A character string specifying the column to use for ordering gene sets in the plot.

Value

A ggplot object representing the dot plot of top pathways.

Examples

```
## Not run:
  # Assuming dose_rider_results is available
  dot_plot <- plot_dotplot_top_pathways(dose_rider_results, top = 10)
  print(dot_plot)

## End(Not run)
```

`plot_gene_random_effect_relationship`

Plot Scatter Plot for Relationship Between Random Intercepts and Effects in a Gene Set

---

Description

This function generates a scatter plot to visualize the relationship between random intercepts and random slopes for genes within a specified gene set. It is particularly useful for analyzing the variability and relationship of gene expressions within a gene set modeled.

Usage

```
plot_gene_random_effect_relationship(dose_rider_results, gene_set_name)
```

Arguments

`dose_rider_results`	A list containing the results from the DoseRider analysis.
`gene_set_name`	The name of the gene set for which the plot will be generated.

Value

A ggplot object representing the scatter plot of random intercepts vs random slopes for the specified gene set. The plot displays the relationship between random intercepts and random slopes for each gene in the gene set.

Examples

```
## Not run:
  # Assuming 'dose_rider_results' contains LMM results for gene sets
 scatter_plot <- plot_gene_random_effect_relationship(dose_rider_results, "Gene Set Name")
  print(scatter_plot)

## End(Not run)
```

---

plot_gene_set_random_effects

Plot Ridge Plots for Random Effects in Gene Sets

Description

This function generates ridge plots to visualize the distribution of random effects for genes within each gene set. It is particularly useful for analyzing the variability of gene expressions within gene sets modeled using Linear Mixed Models (LMMs).

Usage

```
plot_gene_set_random_effects(
  dose_rider_results,
  dose_col = "Dose",
  top = 10,
  order_column = "NegLogPValue",
  decreasing = F
)
```

Arguments

`dose_col`	A character string specifying the name of the dose column in the raw expression data.
`top`	An integer specifying the number of top gene sets to include in the plot. Default is 15.
`order_column`	A character string specifying the column to use for ordering gene sets in the plot

Value

A ggplot object representing the ridge plots of random effects for each gene set. The plot displays the distribution of random effects for each gene set along the y-axis.

Examples

```
## Not run:
  # Assuming 'dose_rider_results' contains LMM results for each gene set
  ridge_plot <- plot_gene_set_random_effects(dose_rider_results)
  print(ridge_plot)

## End(Not run)
```

`plot_pathway_response` *Plot Pathway Response with Enhanced Visualization at Specific Points*

Description

This function creates a plot showing the pathway response with enhanced line thickness at specific zero points of first and second derivatives.

Usage

```
plot_pathway_response(
  dose_rider_results,
  gene_set_name,
  dose_col = "Dose",
  center_values = TRUE,
  scale_values = TRUE,
  legend_position = "none",
  text_size = 4,
  margin_space = 0,
  model_metrics = FALSE,
  v_size = 0.5,
  annotate_gene = FALSE,
  annotation_text_size = 5,
  draw_bmd = TRUE,
  plot_original_data = FALSE,
  clusterResults = TRUE
)
```

Arguments

`dose_rider_results`	A list containing the results from the DoseRider analysis.
`gene_set_name`	The name of the gene set for which to plot the response.

`dose_col`	The name of the column representing dose information. Default is "Dose".
`center_values`	Logical, indicating whether to center the prediction values. Default is TRUE.
`scale_values`	Logical, indicating whether to scale the prediction values. Default is TRUE.
`legend_position`	The position of the legend in the plot. Default is "none".
`text_size`	Numeric, specifying the size of the text in the plot. Default is 4.
`margin_space`	Numeric, specifying the margin space around the plot. Default is 0.
`model_metrics`	Logical, indicating whether to include model metrics in the plot. Default is FALSE.
`v_size`	Numeric, specifying the size of the points where derivatives are zero. Default is 0.5.
`annotate_gene`	Logical, indicating whether to annotate the gene in the plot. Default is FALSE.
`annotation_text_size`	Numeric, specifying the size of the annotation text. Default is 5.
`draw_bmd`	Logical, indicating whether to draw the benchmark dose (BMD) on the plot. Default is TRUE.
`plot_original_data`	Logical, indicating whether to draw original data, or predict the whole range of doses. Default is FALSE.
`clusterResults`	Boolean, if TRUE the genes within a gene set will be clustered to find similar expression patterns. Defaults to TRUE.

Value

A ggplot object representing the pathway response plot.

`plot_tcd_density` *Plot Trend Change Dose (TCD) Density*

Description

This function creates a plot visualizing the density of TCD values and highlights the zero points where the highest density of TCD values are found.

Usage

```
plot_tcd_density(tcd_range_output)
```

Arguments

`tcd_range_output`	A list containing the output from ‘get_tcd_range‘ function, which includes x (TCD values), y (density), and tcd (zero points).

Value

A ggplot object visualizing the density of TCD values with zero points marked.

Examples

```
## Not run:
tcd_range_output <- get_tcd_range(dose_rider_results)
plot_tcd_density_and_zero_points(tcd_range_output)

## End(Not run)
```

---

plot_top_pathway_responses
Plot Top Significant Pathway Responses

Description

Creates a combined plot of top significant pathway responses from dose_rider_results.

Usage

```
plot_top_pathway_responses(
  dose_rider_results,
  top = 6,
  ncol = 3,
  order_column = "NegLogPValue",
  decreasing = F,
  dose_col = "Dose",
  center_values = TRUE,
  scale_values = TRUE,
  legend_position = "none",
  text_size = 4,
  margin_space = 0,
  model_metrics = FALSE,
  v_size = 0.5,
  annotate_gene = FALSE,
  annotation_text_size = 5,
  draw_bmd = TRUE,
  plot_original_data = F,
  clusterResults = F
)
```

Arguments

`dose_rider_results`
A list containing the results from doseRider analysis.

`top`	An integer specifying the number of top gene sets to include in the plot. Default is 15.
`order_column`	A character string specifying the column to use for ordering gene sets in the plot.
`scale_values`	Logical, indicating whether to scale the prediction values. Default is TRUE.
`legend_position`	The position of the legend in the plot.
`text_size`	The size of the text in the plot.
`margin_space`	The margin space around the plot.

Value

A combined ggplot object with top significant pathway response plots.

`prepare_data` *Prepare data for use with glmmTMB from a SummarizedExperiment object*

Description

This function prepares the data from a SummarizedExperiment object into a format suitable for the glmmTMB function. This includes transforming the data to long format, adding size factors and dispersions (if applicable), applying log10 transformation to doses, creating splines for dose, and merging with sample metadata.

Usage

```
prepare_data(
  se,
  geneset,
  dose_col,
  sample_col,
  omic = "rnaseq",
  log_transform = FALSE
)
```

Arguments

`se`	A SummarizedExperiment object containing both assay and metadata.
`geneset`	A character vector of gene names to be considered.
`dose_col`	The name of the dose column in the metadata of the SummarizedExperiment object.
`sample_col`	The name of the sample column in the metadata of the SummarizedExperiment object.
`omic`	A character string specifying the type of omics data. Default is "rnaseq".
`log_transform`	Logical, whether to log10 transform the dose values. Default is FALSE.

Value

A data frame in a format suitable for glmmTMB function.

Examples

```
geneset <- rownames(assay_df)
dose_col <- "condition"
sample_col <- "sample"
long_df <- prepare_data(se, geneset, dose_col, sample_col)
```

`prepare_mean_data` *Prepare mean data and perform centering/standardization if required*

Description

Prepare mean data and perform centering/standardization if required

Usage

```
prepare_mean_data(
  gene_set_results,
  dose_col,
  center_values,
  scale_values,
  clusterResults
)
```

`print.DoseRider` *Print method for DoseRider*

Description

This function prints a summary of a DoseRider object. It shows the class of the object, the number of gene sets it contains, and some details about the first few gene sets.

Usage

```
## S3 method for class 'DoseRider'
print(x, ...)
```

Arguments

`x`	A DoseRider object.
`...`	Further arguments passed to or from other methods.

Examples

```
## Not run:
  # Assuming `dose_rider_result` is a DoseRider object
  print(dose_rider_result)

## End(Not run)
```

`process_gene_set` *Perform DoseRider Analysis Using Linear Mixed Models (LMMs) or Generalized Additive Mixed Models (GAMMs)*

Description

This function performs DoseRider analysis on gene expression data, applying either Linear Mixed Models (LMMs) or Generalized Additive Mixed Models (GAMMs) to each gene set defined in the gene matrix transposed (GMT) format. It evaluates dose-response relationships in the context of gene sets and calculates various model metrics, significance, Benchmark Dose (BMD), and smoothing predictions.

Usage

```
process_gene_set(
  se,
  dose_col,
  sample_col,
  omic,
  gmt,
  i,
  minGSsize = 5,
  maxGSsize = 300,
  covariates = c(),
  modelType = "LMM",
  FilterPathway = FALSE,
  pca_threshold = 0.6,
  models = c("linear", "non_linear_fixed", "non_linear_mixed"),
  log_transform = F
)
```

Arguments

`se`	SummarizedExperiment object or a matrix/data frame containing gene expression data.
`dose_col`	Name of the column representing dose information.
`sample_col`	Name of the column representing sample information.
`omic`	Type of omics data, defaults to "rnaseq".

`gmt`	List of gene sets, each represented as a list with gene names.
`minGSsize`	Minimum gene set size for analysis, defaults to 5.
`maxGSsize`	Maximum gene set size for analysis, defaults to 300.
`covariates`	Optional, vector specifying the covariate column(s) in 'se'.
`modelType`	Type of model to fit for each gene set, options are "LMM" for Linear Mixed Models and "GAMM" for Generalized Additive Mixed Models. Defaults to "LMM".
`FilterPathway`	Boolean, if TRUE the function will apply PCA filtering to detect antagonist patterns. Defaults to FALSE.
`pca_threshold`	Numeric value specifying the variance threshold for PC1 to filter pathways. Default is 0.6.
`log_transform`	Logical, whether to log10 transform the dose values. Default is FALSE.
`method`	Method for multiple testing adjustment, defaults to "fdr".

Value

A list containing results for each gene set including various metrics, p-values, and adjusted p-values. The structure of results will depend on the model type used.

Examples

```
## Not run:
data("SummarizedExperiment")
gmt <- list(geneSet1 = list(genes = c("gene1", "gene2")))
results <- DoseRider(se, gmt, "dose", "sample", "covariate", "rnaseq", modelType = "GAMM")

## End(Not run)
```

`read_gmt` *Read GMT File from MSigDB*

Description

Parses a GMT (Gene Matrix Transposed) file from the Molecular Signatures Database (MSigDB), structuring it into a list of pathways and their associated genes. Each element of the list is a list itself, containing the pathway name and a vector of gene symbols.

Usage

```
read_gmt(file_path)
```

Arguments

`file_path`	The path to the GMT file to be read.

Value

A list where each element is a list with two elements: ‘$pathway‘, the name of the pathway, and ‘$genes‘, a vector of gene symbols associated with that pathway.

Examples

```
file_path <- "path/to/your/c2.cgp.v2023.2.Hs.symbols.gmt"
gmt_data <- read_gmt(file_path)
if (length(gmt_data) > 0) {
  print(gmt_data[[1]])
  print(gmt_data[[2]])
}
```

`select_best_model` *Select Best Model Based on AICc and P-Values*

Description

This function selects the best model from a list of models (null, linear, non-linear fixed, and non-linear mixed) by comparing their AICc values and significance of p-values against the null model. The best model is the one with the lowest AICc and a significant p-value.

Usage

```
select_best_model(model_list, p_value_list, alpha = 0.05)
```

Arguments

`model_list`	A named list of model objects. Names should include ’null’, ’linear’, ’non_linear_fixed’, and ’non_linear_mixed’.
`p_value_list`	A named list of p-values for model comparisons against the null model.
`alpha`	Significance level for p-value comparison (default is 0.05).

Value

The name of the best fitting model (’null’, ’linear’, ’non_linear_fixed’, or ’non_linear_mixed’).

Examples

```
## Not run:
# Assuming you have a list of models called 'model_list'
# and a list of p-values called 'p_value_list':
best_model <- select_best_model(model_list, p_value_list)

## End(Not run)
```

`smooth_pathway_trend` *Generate Smooth Predictions for Pathway Trends*

Description

This function creates a data frame of smooth predictions based on a provided model. It is useful for visualizing trends in pathway data, especially in the context of dose-response studies. It handles different types of omics data and can incorporate covariates.

Usage

```
smooth_pathway_trend(
  model,
  long_df,
  dose_col = "dose",
  sample_col = "sample",
  omic = "rnaseq",
  random_effects = TRUE,
  covariates_cols = NULL,
  gene_subset = NULL,
  dose_points = 25,
  sample_subset_size = 10
)
```

Arguments

`model`	A model object, typically of class 'lmerMod' or 'glmerMod'.
`long_df`	A data frame containing the input data with columns corresponding to dose, sample, and other variables.
`dose_col`	The name of the dose variable in 'long_df'.
`sample_col`	The name of the sample variable in 'long_df'.
`omic`	A character string indicating the type of omics data (e.g., "rnaseq").
`random_effects`	Logical, indicating if random effects (like gene) should be included.
`covariates_cols`	Optional; a vector of names of additional covariates in 'long_df'.

Value

A data frame containing the original data along with the predictions from the model.

Examples

```
## Not run:
data("mtcars")
model <- lmer(mpg ~ wt + (1|cyl), data = mtcars)
predictions <- smooth_pathway_trend(model = model,
                                   long_df = mtcars,
                                   dose_col = "wt",
                                   sample_col = "cyl",
                                   omic = "rnaseq",
                                   random_effects = TRUE,
                                   covariates_cols = NULL)

## End(Not run)
```

---

transform_smooth_values

Transform Smooth Values to Gene vs. Dose Dataframe

Description

This function transforms smooth values into a dataframe with genes as rows and doses as columns. Each row (gene) is centered and scaled (standardized) to normalize the data.

Usage

```
transform_smooth_values(
  smooth_values,
  dose_col = "Dose",
  expression_col = "predictions",
  center_values = TRUE,
  scale_values = TRUE
)
```

Arguments

`smooth_values`	Dataframe containing smooth values from doseRider results.
`dose_col`	Character string specifying the name of the dose column in ‘smooth_values‘.
`expression_col`	Character string specifying the name of the expression column in ‘smooth_values‘.
`center_values`	Logical, whether to center the gene values (default is TRUE).
`scale_values`	Logical, whether to scale the gene values (default is TRUE).

Value

A transformed dataframe with genes as rows, doses as columns, and each row centered and scaled.

Examples

```
## Not run:
  # Assuming 'smooth_values' is available
  transformed_data <- transform_smooth_values(smooth_values, dose_col = "Dose")

## End(Not run)
```

Index