Package 'MetabolomicsPipeline'

April 29, 2025

Title Metabolomics Pipeline Tools

Version 0.99.0

Description This package provides analysis tools for analyzing metabolomics data. We provide functionality for hypothesis testing at
the subpathway level, pairwise comparisons of metabolites, and tools for exploratory analysis.

biocViews Metabolomics, Software

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests openxlsx, rmarkdown, table1, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports dplyr (>= 1.1.3), emmeans (>= 1.8.8), factoextra (>= 1.0.7), FactoMineR (>= 2.8), ggplot2 (>= 3.4.3), ggplotify, grDevices (>= 3.6.0), kableExtra (>= 1.3.4), knitr (>= 1.44), magrittr, methods (>= 4.3.1), pheatmap (>= 1.0.12), plotly (>= 4.10.2), RColorBrewer (>= 1.1.3), readxl, reshape2 (>= 1.4.4), S4Vectors, stats, stringr (>= 1.5.0), SummarizedExperiment, tibble, tidyr (>= 1.3.0)

Depends R (>= 4.4.0)

VignetteBuilder knitr

URL https://github.com/datalifecycle-ua/MetabolomicsPipeline

BugReports https://github.com/datalifecycle-ua/MetabolomicsPipeline/issues

LazyData true

NeedsCompilation no

Author Joel Parker [aut, cre] (ORCID: <https://orcid.org/0000-0002-3411-3818>), Bonnie LaFleur [aut]

Maintainer Joel Parker <joelparker@arizona.edu>

Contents

`MetabolomicsPipeline-package`
MetabolomicsPipeline: Metabolomics Pipeline Tools

Description

This package provides analysis tools for analyzing metabolomics data. We provide functionality for hypothesis testing at the subpathway level, pairwise comparisons of metabolites, and tools for exploratory analysis.

Author(s)

Maintainer: Joel Parker <joelparker@arizona.edu> (ORCID)

Authors:

- Bonnie LaFleur <blafleur@arizona.edu>

See Also

Useful links:

- https://github.com/datalifecycle-ua/MetabolomicsPipeline
- Report bugs at https://github.com/datalifecycle-ua/MetabolomicsPipeline/issues

all_sig_subpath *Table of Significant Subpathways*

Description

Create a table of all significant subpathways

Usage

```
all_sig_subpath(path_results)
```

Arguments

path_results	Results data frame generated by subpathway_analysis

Value

A table of all significant subpathways. Including the significant model type and model type p-value.

Examples

```
# Load data
data("demoDataSmall", package = "MetabolomicsPipeline")
dat <- demoDataSmall

# Runsubpathay analysis
sub_analysis <- subpathway_analysis(dat,
    treat_var = "GROUP_NAME",
    block_var = "TIME1",
    strat_var = NULL,
    Assay = "normalized"
)

##############################################################################
### Results Plots ############################################################
##############################################################################

# significant subpathways by model type
subpath_by_model(sub_analysis)

# Percentage of signficant subpathways within superpathways
subpath_within_superpath(sub_analysis)

met_within_sub(sub_analysis, subpathway = "Aminosugar Metabolism")

# All signifiicant subpathways
all_sig_subpath(sub_analysis)
```

create_heatmap_Data *Create metadata and matricies for metabolite heatmaps*

Description

This function creates the required matrices for the metabolite heatmaps.

Usage

```
create_heatmap_Data(data, heatmap_variables, Assay = "normalized", ...)
```

Arguments

data	A SummarizedExperiment containing metabolomics data.
heatmap_variables	A vector of variable names that are NOT metabolites.
Assay	Name of assay data to be used for heatmaps. Default="normalized".
...	Additional arguments that can be passed into the arrange function. This parameter will order the columns of the heatmap data.

Value

A list of matrices including the heatmap variable (meta data for heatmap) and the values for the heatmap.

create_met_se *Create a SummarizedExperiment from Metabolomics Data*

Description

This function constructs a `SummarizedExperiment` object from metabolomics data. It uses peak intensity data, sample metadata as `colData`, and chemical annotations as `rowData`. All inputs must be aligned via identifiers (sample_names and chemicalID)

Usage

```
create_met_se(
  chemical_annotation,
  sample_metadata,
  peak_data,
  sample_names = "PARENT_SAMPLE_NAME",
  chemical_id = "CHEM_ID"
)
```

Arguments

chemical_annotation	A data.frame where each row represents a chemical.
sample_metadata	A data.frame where each row represents a biological sample.
peak_data	A data.frame with samples as rows and chemicals as columns. Sample names must be the first column.
sample_names	Column name in the meta data containing the sample names. This must correspond to the row names of the raw peak data in the excel file.
chemical_id	Column name in the meta data containing the sample names. This must correspond to the column names of the raw peak data.

Value

A SummarizedExperiment object containing:

- assays: the peak data
- rowData: the chemical annotation
- colData: the sample metadata

Examples

```
#Get demo sample metadata
data("demoSampleMeta", package = "MetabolomicsPipeline")

#Get demo chemical annotation file
data("demoChemAnno", package = "MetabolomicsPipeline")

# Get demo peak data
data("demoPeak", package = "MetabolomicsPipeline")

dat <- create_met_se(chemical_annotation = demoChemAnno,
                     sample_metadata = demoSampleMeta,
                     peak_data = demoPeak,
                     chemical_id = "CHEM_ID",
                     sample_names = "PARENT_SAMPLE_NAME")
```

---

demoChemAnno *Demo Chemical Annotation Data*

Description

A small example of chemical annotation data for the MetabolomicsPipeline package.

Usage

```
demoChemAnno
```

Format

A data frame with annotation for 1102 metabolites

Source

Generated for demonstration

`demoDat` *Demo data for the MetabolomicsPipeline,*

Description

Demo data consisting of 86 samples (42 males, 44 females), three treatment groups, and the samples were taken

Usage

```
data(demoDat)
```

Format

SummarizedExperiment object

Value

A SummarizedExperiment object with 86 samples

`demoDataSmall` *Subset of Demo data for the MetabolomicsPipeline ,*

Description

Demo data consisting of 86 samples (42 males, 44 females), three treatment groups, and the samples were taken at three different time points. We focus on a subset of 10 Subpathways.

Usage

```
data(demoDataSmall)
```

Format

Rd

Value

A subset of the metabolites in the DemoData.

demoPeak *Demo peak data for the MetabolomicsPipeline,*

Description

Peak data for demoDat.

Usage

```
data(demoPeak)
```

Format

data.frame

Value

A dataframe with peak data for demoDat

demoSampleMeta *Demo sample metadata for the MetabolomicsPipeline,*

Description

Demo data consisting of 86 samples (42 males, 44 females), three treatment groups, and the samples were taken

Usage

```
data(demoSampleMeta)
```

Format

data.frame

Value

A dataframe with metadata for with 86 samples.

load_met_excel *Load Metabolomic Data as SummarizedExperiment*

Description

Automatically load metabolomic data from excel file

Usage

```
load_met_excel(
  path,
  raw_sheet = "Peak Area Data",
  chemical_sheet = "Chemical Annotation",
  sample_meta = "Sample Meta Data",
  normalized_peak = "Log Transformed Data",
  sample_names = "PARENT_SAMPLE_NAME",
  chemicalID = "CHEM_ID"
)
```

Arguments

path	Path to excel file with peak data, chemical annotations, sample meta data, and (optionally) the normalized peak counts
raw_sheet	Sheet name for the raw peak data.
chemical_sheet	Sheet name for chemical annotation.
sample_meta	Sheet name for sample meta data.
normalized_peak	Sheet name for the normalized peak data. If you are not adding the normalized data from the excel file then set normalized_peak=NA.
sample_names	Column name in the meta data containing the sample names. This must correspond to the row names of the raw peak data in the excel file.
chemicalID	Column name in the meta data containing the sample names. This must correspond to the column names of the raw peak data.

Details

The metabolomics experiment data are stored in a SummarizedExperiment.

Value

A SummarizedExperiment containing metabolomics expirement data.

See Also

SummarizedExperiment::SummarizedExperiment

log_transformation *Log Transformation of Metabolite Data*

Description

This function performs a natural logarithm (log) transformation on metabolite data stored within a SummarizedExperiment object. All numeric metabolite values are log-transformed, and the resulting data are added as a new assay named "normalized".

Usage

```
log_transformation(met_se, assay = "min_impute")
```

Arguments

met_se	A SummarizedExperiment object containing metabolite data.
assay	A character string specifying the assay name within met_se to log-transform. Default is "min_impute".

Value

The input SummarizedExperiment object with a new assay "normalized" containing the log-transformed data.

Examples

```
library(SummarizedExperiment)
data("demoDataSmall", package = "MetabolomicsPipeline")

# Median standardization
demoDataSmall <- median_standardization(met_se = demoDataSmall,
 assay = "peak")

# Minimum value imputation
demoDataSmall <- min_val_impute(met_se = demoDataSmall, assay = "median_std")

# Log transformation
demoDataSmall <- log_transformation(met_se = demoDataSmall,
 assay = "min_impute")

# Access log-transformed data
assay(demoDataSmall, "normalized")[1:5, 1:5]
```

median_standardization *Median standardization for metabolite data*

Description

This function performs median standardization of metabolite data stored within a SummarizedExperiment object. For each metabolite, the values are divided by the median value of that metabolite across samples. The standardized data are returned as a data frame and also added to the original SummarizedExperiment object as a new assay named "median_std".

Usage

```
median_standardization(met_se, assay = "peak")
```

Arguments

met_se	A SummarizedExperiment object containing metabolite data.
assay	A character string specifying the assay name within met_se to be median standardized. Default is "peak".

Value

A data frame of median standardized metabolite data. The SummarizedExperiment object is also updated internally with a new assay "median_std".

Examples

```
library(SummarizedExperiment)
data("demoDataSmall", package = "MetabolomicsPipeline")

# Median standardization
peak_med <- median_standardization(met_se = demoDataSmall, assay = "peak")

# Access the median standardized data within the SummarizedExperiment
assay(peak_med, "median_std")[1:5, 1:5]
```

metabolite_heatmap *Create metabolite heatmap*

Description

Create heatmaps which are arranged by the experimental conditions.

Usage

```
metabolite_heatmap(
  data,
  top_mets = 50,
  group_vars,
  strat_var = NULL,
  caption = NULL,
  Assay = "normalized",
  sample_names = "PARENT_SAMPLE_NAME",
  ...
)
```

Arguments

data	A SummarizedExperiment containing the metabolomics experiment data.
top_mets	Number of metabolites to include in the heatmap. Metabolites are chosen based on the highest variability.
group_vars	Vector of variables to annotate heatmap with. Columns will be grouped by these variables.
strat_var	Variable to stratify the heatmap by.
caption	A title for the heatmap. If strat_var is used, the title will automatically include the stratum with the tile.
Assay	Which assay data to use for the heatmap (default="normalized").
sample_names	Column name in the meta data containing the sample names. This must correspond to the row names of the raw peak data in the excel file.
...	Additional arguments can be passed into the arrange function. This parameter will order the columns of the heatmap.

Value

A gtable class with all of the information to build the heatmap. To view the heatmap use ggplotify::as.ggplot().

Examples

```
# load data
data("demoDat", package = "MetabolomicsPipeline")
dat <- demoDat

# Heatmap with one group
treat_heatmap <- metabolite_heatmap(dat,
    top_mets = 50,
    group_vars = "GROUP_NAME",
    strat_var = NULL,
    caption = "Heatmap Arranged By Group",
    Assay = "normalized",
    GROUP_NAME
)
```

`metabolite_pairwise` *Metabolite Pairwise Comparisons.*

Description

Computes the pairwise comparison estimates and p-values for each metabolite.

Usage

```
metabolite_pairwise(
  data,
  form,
  Assay = "normalized",
  strat_var = NULL,
  mets = NULL
)
```

Arguments

`data`	SummarizedExperiment with metabolomics experiment data.
`form`	This is a character string the resembles the right hand side of a simple linear regression model in R. For example form = "Group1 + Group2".
`Assay`	Name of the assay to be used for the pairwise analysis (default='normalized')
`strat_var`	A variable in the analysis data to stratify the model by. If this is specified, a list of results will be returned.
`mets`	Chemical ID for the metabolites of interest. If NULL then the pairwise analysis is completed for all metabololites.

Details

This function will analyze each metabolite individually. For each metabolite, the metabolite_pairwise function will first test whether the model explained a significant proportion of the variance in the metabolite using an F-test. Since we will be looking at multiple comparisons for the metabolite, it is good practice to first look at the overall p-value from the F-test before looking at the pairwise comparisons. The metabolite_pairwise function then looks at all pairwise comparisons utilizing the emmeans package. The metabolite_pairwise function returns a data frame with the metabolite overall p-value, log fold change for each group, and the p-value for each comparison.

Value

The overall F-test p-value, and the estimate and pvalue for each pairwise comparison.

Examples

```
# Load data
data("demoDataSmall", package = "MetabolomicsPipeline")
dat <- demoDataSmall

# Run pairwise analysis
strat_pairwise <- metabolite_pairwise(dat,
    form = "GROUP_NAME*TIME1",
```

```
    strat_var = "Gender"
)

##############################################################################
## Create Estimate Heatmap ###################################################
##############################################################################

met_est_heatmap(strat_pairwise$Female, dat,
                interactive = FALSE,
                CHEM_ID = "CHEM_ID", SUB_PATHWAY = "SUB_PATHWAY",
                CHEMICAL_NAME = "CHEMICAL_NAME",
                main = "Log fold change heatmap", show_rownames = FALSE
)


##############################################################################
## Create P-value Heatmap ####################################################
##############################################################################
# Female
met_p_heatmap(strat_pairwise$Female, dat,
              interactive = FALSE, show_rownames = FALSE,
              main = "Pvalue Heatmap"
)
```

`metabolite_pca` *Metabolite PCA*

Description

Computes and plots the first two components of the PCA from the metabolite data.

Usage

```
metabolite_pca(data, Assay = "normalized", meta_var)
```

Arguments

`data`	SummarizedExperiment with metabolomics experiment data.
`Assay`	Name of the assay to be used for the pairwise analysis (default='normalized')
`meta_var`	A metadata variable to color code the PCA plot by.

Value

A PCA plot of the first two principal components, colored by the metadata variable.

Examples

```
# load data
data("demoDat", package = "MetabolomicsPipeline")
dat <- demoDat

# Define PCA label from metadata
```

```
meta_var <- "Gender"

# Run PCA
pca <- metabolite_pca(dat,
    meta_var = meta_var
)


# Show PCA
pca
```

`met_est_heatmap` *Metabolite Pairwise Estimate Interactive Heatmap.*

Description

Produce an interactive heatmap of the estimates produced in `metabolite_pairwise`.

Usage

```
met_est_heatmap(
  results_data,
  data,
  interactive = FALSE,
  SUB_PATHWAY = "SUB_PATHWAY",
  CHEMICAL_NAME = "CHEMICAL_NAME",
  plotlyTitle = "Metabolite log fold change",
  ...
)
```

Arguments

`results_data`	Results data frame of the pairwise comparisons produced by `metabolite_pairwise`.
`data`	A SummarizedExperiment containing the metabolomics experiment data.
`interactive`	boolean (T/F) for whether or not the plot should be interactive. Use interactive=T to produce an interactive plot using plotly. Use interactive=F to produce a static heatmap using pheatmap.
`SUB_PATHWAY`	Column name in the chemical annotation worksheet which contains the subpathway information.
`CHEMICAL_NAME`	Column name in the chemical annotation worksheet which contains the chemical name.
`plotlyTitle`	Title for the interactive heatmap.
`...`	Additional arguments that can be passed to pheatmap.

Details

This function will produce a heatmap of the log fold changes for the metabolites with a significant overall p-value (which tested if the treatment group means were equal under the null hypothesis). The heatmap colors will only show if the log fold-change is greater than log(2) or less than log(.5). Therefore, this heatmap will only focus on comparisons with a fold change of two or greater.

Value

An interactive heatmap of pairwise estimates.

Examples

```
# Load data
data("demoDataSmall", package = "MetabolomicsPipeline")
dat <- demoDataSmall

# Run pairwise analysis
strat_pairwise <- metabolite_pairwise(dat,
    form = "GROUP_NAME*TIME1",
    strat_var = "Gender"
)

##############################################################################
## Create Estimate Heatmap ###################################################
##############################################################################

met_est_heatmap(strat_pairwise$Female, dat,
                interactive = FALSE,
                 SUB_PATHWAY = "SUB_PATHWAY",
                CHEMICAL_NAME = "CHEMICAL_NAME",
                plotlyTitle = "Metabolite log fold change",
                main = "Log fold change heatmap", show_rownames = FALSE
)


##############################################################################
## Create P-value Heatmap ####################################################
##############################################################################
# Female
met_p_heatmap(strat_pairwise$Female, dat,
              interactive = FALSE, show_rownames = FALSE,
              plotlyTitle = "P-Value Heatmap",
              main = "Pvalue Heatmap"
)
```

`met_p_heatmap` *Metabolite Pairwise P-Value Interactive Heatmap.*

Description

Produce an interactive heatmap of the p-values produced in `metabolite_pairwise`.

Usage

```
met_p_heatmap(
  results_data,
  data,
  interactive = FALSE,
```

```
  SUB_PATHWAY = "SUB_PATHWAY",
  CHEMICAL_NAME = "CHEMICAL_NAME",
  plotlyTitle = "P-Value Heatmap",
  ...
)
```

Arguments

results_data	Results data frame of the pairwise comparisons produced by metabolite_pairwise.
data	A SummarizedExperiment containing metabolomics experiment data.
interactive	boolean (T/F) for whether or not the plot should be interactive. Use interactive=T to produce an interactive plot using plotly. Use interactive=F to produce a static heatmap using pheatmap.
SUB_PATHWAY	Column name in the chemical annotation worksheet which contains the subpathway information.
CHEMICAL_NAME	Column name in the chemical annotation worksheet which contains the chemical name.
plotlyTitle	Title for the interactive heatmap.
...	Additional arguments that can be passed to pheatmap.

Details

For the metabolites which had a significant overall p-value (which tested if the treatment group means were equal under the null hypothesis), we will produce a heatmap of the p-values.

Value

An interactive heatmap of pairwise p-values.

Examples

```
# Load data
data("demoDataSmall", package = "MetabolomicsPipeline")
dat <- demoDataSmall

# Run pairwise analysis
strat_pairwise <- metabolite_pairwise(dat,
    form = "GROUP_NAME*TIME1",
    strat_var = "Gender"
)

##############################################################################
## Create Estimate Heatmap ###################################################
##############################################################################

met_est_heatmap(strat_pairwise$Female, dat,
                interactive = FALSE,
                CHEM_ID = "CHEM_ID", SUB_PATHWAY = "SUB_PATHWAY",
                CHEMICAL_NAME = "CHEMICAL_NAME",
                plotlyTitle = "Estimate Heatmap",
                main = "Log fold change heatmap", show_rownames = FALSE
)
```

```
##############################################################################
## Create P-value Heatmap ####################################################
##############################################################################
# Female
met_p_heatmap(strat_pairwise$Female, dat,
              interactive = FALSE, show_rownames = FALSE,
              plotlyTitle = "P-Value Heatmap",
              main = "Pvalue Heatmap"
)
```

`met_within_sub` *Metabolites within Subpathway Table*

Description

Return the model results for each metabolite within a subpathway.

Usage

```
met_within_sub(
  subpath_results,
  subpathway,
  mod = c("interaction", "parallel", "single")
)
```

Arguments

`subpath_results`	Results data frame generated by `subpathway_analysis`
`subpathway`	Character string of the subpathway of interest. This is case sensitive and must be listed in the subpath_results.
`mod`	Model of interest. This can be a single model or a vector of model types that can take on the values "interaction", "parallel", or "single".

Value

A table with the results from the model types specified and for each metabolite within the superpathway specified.

Examples

```
data("demoDataSmall", package = "MetabolomicsPipeline")
dat <- demoDataSmall

# Runsubpathay analysis
sub_analysis <- subpathway_analysis(dat,
    treat_var = "GROUP_NAME",
    block_var = "TIME1",
    strat_var = NULL,
```

```
    Assay = "normalized"
)

##############################################################################
### Results Plots ############################################################
##############################################################################

# significant subpathways by model type
subpath_by_model(sub_analysis)

# Percentage of signficant subpathways within superpathways
subpath_within_superpath(sub_analysis)

met_within_sub(sub_analysis, subpathway = "Aminosugar Metabolism")
```

`min_val_impute` *Minimum Value Imputation for Metabolite Data*

Description

This function imputes missing values in metabolite data stored within a `SummarizedExperiment` object. For each metabolite, missing values are replaced with the minimum observed value for that metabolite. The imputed data are added to the `SummarizedExperiment` object as a new assay named `"min_impute"`.

Usage

```
min_val_impute(met_se, assay = "median_std")
```

Arguments

`met_se`	A `SummarizedExperiment` object containing metabolite data.
`assay`	A character string specifying the assay name within `met_se` to perform minimum value imputation on. Default is `"median_std"`.

Value

The input `SummarizedExperiment` object with a new assay `"min_impute"` containing the minimum value-imputed data.

Examples

```
library(SummarizedExperiment)
data("demoDataSmall", package = "MetabolomicsPipeline")

# Median standardization
demoDataSmall <- median_standardization(met_se = demoDataSmall,
 assay = "peak")

# Minimum value imputation
demoDataSmall <- min_val_impute(met_se = demoDataSmall, assay = "median_std")
```

```
# Access the imputed data
assay(demoDataSmall, "min_impute")[1:5, 1:5]
```

`pairwise` *Pairwise function*

Description

This is the main function for metabolite_pairwise

Usage

```
pairwise(out, form, data)
```

Arguments

`out`	Outcome used as reponse
`form`	form of the model
`data`	data used for modeling

Value

Pairwise comparisons for a single metabolite.

`subpathway_analysis` *Subpathway Analysis*

Description

Subpathway analysis for metabolite data.

Usage

```
subpathway_analysis(
  data,
  treat_var,
  block_var = NULL,
  strat_var = NULL,
  Assay = "normalized",
  subPathwayName = "SUB_PATHWAY",
  chemName = "CHEMICAL_NAME",
  superPathwayName = "SUPER_PATHWAY"
)
```

Arguments

data	SummarizedExperiment with metabolomics experiment data.
treat_var	This is the name of the variable in the analysis data that is the main variable of interest.
block_var	This is the name of the blocking variable in the dataset. If the the experimental design does not include a blocking variable, then the value of block_var=NULL.
strat_var	Variable to stratify the subpathway analysis by. This is set to NULL by default and will not stratify the analysis unless specified.
Assay	Name of the assay to be used for the pairwise analysis (default='normalized')
subPathwayName	Column name for subpathway variable as defined in the chemical annotation worksheet.
chemName	Column name for chemical name variable as defined in the chemical annotation worksheet.
superPathwayName	Column name for super-pathway variable as defined in the chemical annotation worksheet.

Details

For each metabolite, we test three models using using ANOVA.

1. Interaction: $logPeak = Treatment + block + Treatment * block$
2. Parallel: $logPeak = Treatment + block$
3. Single: $logPeak = Treatment$

For the interaction model, we are focusing only on the interaction term "Treatment*block" to test if there is a significant interaction between our treatment and the block variable. The parallel model tests if the block variable explains a significant amount of the metabolite variance, and the treatment model tests if the treatment explains a significant proportion of the variance for each metabolite. Then, we use the Combined Fisher probability to test each model at the subpathway level.

$$\tilde{X} = -2\sum_{i=1}^{k} ln(p_i)$$

where k is the number of metabolites in the subpathway. We can get a p-value from $P(X \geq \tilde{X})$, knowing that $\tilde{X} \sim \chi^2_{2k}$. You will notice that smaller p-values will lead to a larger $\tilde{X}$.

Value

A data frame with "CHEM_ID","sub_pathway","chem_name", "interaction_pval","interaction_fisher","parallel_pval","p
"single_pval","single_fisher",and "model" for each metabolite.

See Also

Loughin, Thomas M. "A systematic comparison of methods for combining p-values from independent tests." Computational statistics & data analysis 47.3 (2004): 467-485.

Examples

```
# Load data
data("demoDataSmall", package = "MetabolomicsPipeline")
dat <- demoDataSmall

# Runsubpathay analysis
sub_analysis <- subpathway_analysis(dat,
    treat_var = "GROUP_NAME",
    block_var = "TIME1",
    strat_var = NULL,
    Assay = "normalized"
)

##############################################################################
### Results Plots ############################################################
##############################################################################

# significant subpathways by model type
subpath_by_model(sub_analysis)

# Percentage of signficant subpathways within superpathways
subpath_within_superpath(sub_analysis)

met_within_sub(sub_analysis, subpathway = "Aminosugar Metabolism")

# All signifiicant subpathways
all_sig_subpath(sub_analysis)
```

`subpathway_boxplots` *Subpathway Boxplots*

Description

Creates boxplots for each metabolite within a specified subpathway.

Usage

```
subpathway_boxplots(
  data,
  subpathway,
  block_var,
  treat_var,
  Assay = "normalized",
  CHEMICAL_NAME = "CHEMICAL_NAME",
  SUB_PATHWAY = "SUB_PATHWAY",
  ...
)
```

Arguments

`data`	SummarizedExperiment with metabolomics experiment data.

subpathway	Character value of the subpathway of interest. This is case sensitive and must be in the chemical annotation file.
block_var	This the the name of the variable in the meta data that is used for the X axis of the box plots. We recommend using the "block_var" from the subpathway analysis.
treat_var	This is a grouping variable. As a recommendation the treatment groups should be used in the treat_var argument as this will provide a different color for each of the treatments making it easier to identify.
Assay	Name of the assay to be used for the pairwise analysis (default='normalized')
CHEMICAL_NAME	Column name in the chemical annotation worksheet which contains the chemical name.
SUB_PATHWAY	Column name in chemical annotation file which contains the SUB_PATHWAY information
...	Additional arguments to filter the analysis data by.

Details

.

Value

Boxplots stratified by metabolites.

Examples

```
# load data
data("demoDat", package = "MetabolomicsPipeline")
dat <- demoDat

################################################################################
### BoxPlots ###################################################################
################################################################################

subpathway_boxplots(dat,
    subpathway = "Lactoyl Amino Acid", block_var = TIME1,
    treat_var = GROUP_NAME, Assay = "normalized",
    CHEMICAL_NAME = "CHEMICAL_NAME",
    SUB_PATHWAY = "SUB_PATHWAY", Gender == "Female"
)


################################################################################
## Line plots ##################################################################
################################################################################

# Set up data
dat$TIME1 <- as.numeric(factor(dat$TIME1,
    levels = c("PreSymp", "Onset", "End")
))
# Create line plots
subpathway_lineplots(dat,
                     subpathway = "Lactoyl Amino Acid",
                     block_var = TIME1, treat_var = GROUP_NAME,
                     Assay = "normalized",
```

```
                    CHEMICAL_NAME = "CHEMICAL_NAME",
                    SUB_PATHWAY="SUB_PATHWAY",Gender == "Female")
```

`subpathway_lineplots` *Subpathway Lineplots*

Description

Create line plots for each metabolite within a subpathway.

Usage

```
subpathway_lineplots(
  data,
  subpathway,
  block_var,
  treat_var,
  Assay = "normalized",
  CHEMICAL_NAME = "CHEMICAL_NAME",
  SUB_PATHWAY = "SUB_PATHWAY",
  ...
)
```

Arguments

`data`	SummarizedExperiment with metabolomics experiment data.
`subpathway`	Character value of the subpathway of interest. This is case sensitive and must be in the chemical annotation file.
`block_var`	This the the name of the variable in the meta data that is used for the X axis of the line plots. We recommend using the "block_var" variable from the subpathway analyis.
`treat_var`	This is a grouping variable. As a recommendation the treatment groups should be used in the groupBy argument as this will provide a different color for each of the treatments making it easier to identify.
`Assay`	Name of the assay to be used for the pairwise analysis (default='normalized')
`CHEMICAL_NAME`	Column name in the chemical annotation worksheet which contains the chemical name.
`SUB_PATHWAY`	Column name in the chemical annotation worksheet which contains the subpathway information.
`...`	Additional arguments to filter the analysis data by.

Value

Line plots stratified by metabolite.

Examples

```
data("demoDat", package = "MetabolomicsPipeline")
dat <- demoDat

################################################################################
### BoxPlots ###################################################################
################################################################################

subpathway_boxplots(dat,
    subpathway = "Lactoyl Amino Acid", block_var = TIME1,
    treat_var = GROUP_NAME, Assay = "normalized",
    CHEMICAL_NAME = "CHEMICAL_NAME",
    SUB_PATHWAY = "SUB_PATHWAY", Gender == "Female"
)


################################################################################
## Line plots ##################################################################
################################################################################

# Set up data
dat$TIME1 <- as.numeric(factor(dat$TIME1,
    levels = c("PreSymp", "Onset", "End")
))

# Create line plots
subpathway_lineplots(dat,
    subpathway = "Lactoyl Amino Acid",
    block_var = TIME1, treat_var = GROUP_NAME,
    Assay = "normalized",
    CHEMICAL_NAME = "CHEMICAL_NAME",
    SUB_PATHWAY = "SUB_PATHWAY", Gender == "Female"
)
```

---

subpath_by_model *Subpathway model type table*

Description

Create a table with the number of significant subpathways for each model type.

Usage

```
subpath_by_model(subpath_results)
```

Arguments

`subpath_results`
: Results data frame generated by `subpathway_analysis`

Details

Each subpathway will only have one model type. We first test the interaction, and then the parallel and single models are tested last. Suppose a subpathway has a significant interaction model type. In that case, the table will count it as an interaction and not as a parallel or single.

Value

A table of the number of significant subpathways by model type.

Examples

```
# Load data
data("demoDataSmall", package = "MetabolomicsPipeline")
dat <- demoDataSmall

# Runsubpathay analysis
sub_analysis <- subpathway_analysis(dat,
    treat_var = "GROUP_NAME",
    block_var = "TIME1",
    strat_var = NULL,
    Assay = "normalized"
)

################################################################################
### Results Plots ##############################################################
################################################################################

# significant subpathways by model type
subpath_by_model(sub_analysis)

# Percentage of signficant subpathways within superpathways
subpath_within_superpath(sub_analysis)

met_within_sub(sub_analysis, subpathway = "Aminosugar Metabolism")
```

---

subpath_within_superpath

Proportion of the Significant Subpathways Within Superpathways

---

Description

Create a table that gives the percentage of significant subpathways within each superpathway.

Usage

```
subpath_within_superpath(subpath_results)
```

Arguments

subpath_results	Results data frame generated by subpathway_analysis

Value

A table with the proportion (and percent) of significant subpathways within superpathways.

Examples

```
# Load data
data("demoDataSmall", package = "MetabolomicsPipeline")
dat <- demoDataSmall

# Runsubpathay analysis
sub_analysis <- subpathway_analysis(dat,
    treat_var = "GROUP_NAME",
    block_var = "TIME1",
    strat_var = NULL,
    Assay = "normalized"
)

################################################################################
### Results Plots ##############################################################
################################################################################

# significant subpathways by model type
subpath_by_model(sub_analysis)

# Percentage of signficant subpathways within superpathways
subpath_within_superpath(sub_analysis)

met_within_sub(sub_analysis, subpathway = "Aminosugar Metabolism")
```