Package 'tidyexposomics'

May 4, 2026

Title Integrated Exposure-Omics Analysis Powered by Tidy Principles

Version 1.1.0

Description The tidyexposomics package is designed to facilitate the integration of exposure and omics data to identify exposure-omics associations. We structure our commands to fit into the tidyverse framework, where commands are designed to be simplified and intuitive. Here we provide functionality to perform quality control, sample and exposure association analysis, differential abundance analysis, multi-omics integration, and functional enrichment analysis.

License MIT + file LICENSE

URL https://bionomad.github.io/tidyexposomics/

BugReports https://github.com/BioNomad/tidyexposomics/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

biocViews Software, Transcriptomics, GeneExpression, Epigenetics, Proteomics, DifferentialExpression, DifferentialMethylation, QualityControl, GraphAndNetwork, MultipleComparison, Regression, StatisticalMethod, Visualization, WorkflowStep

Imports BiocFileCache, broom, cluster, dplyr, DT, factoextra, fenr, ggplot2 (>= 3.4.0), ggpubr, ggrepel, Hmisc, httr, igraph, jsonlite, limma, MASS, methods, mixOmics, naniar, purrr, readr, RGCCA, rlang, S4Vectors, scales, shiny, stats, stringr, SummarizedExperiment, tibble, tidybulk, tidyr, utils

Depends R (>= 4.5.0), MultiAssayExperiment

Suggests BiocStyle, circlize, curl, densityClust, DiagrammeR, dynamicTreeCut, edgeR, forcats, ggh4x, ggnewscale, ggraph, ggridges, ggsci, ggvenn, grid, gridExtra, impute, janitor, knitr, matrixStats, mice, mirt, missForest, MOFA2, nipalsMCIA, openxlsx, patchwork, reticulate, rmarkdown, testthat (>= 3.0.0), tidygraph, tidyHeatmap, tidytext, tidyverse

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/tidyexposomics

git_branch devel

git_last_commit 7d197e3

git_last_commit_date 2026-04-28

Repository Bioconductor 3.24

Date/Publication 2026-05-03

Author Jason Laird [aut, cre] (ORCID: <https://orcid.org/0009-0000-5994-2236>), Thomas Hartung [ctb] (ORCID: <https://orcid.org/0000-0003-1359-7689>), Fenna Sillé [ctb] (ORCID: <https://orcid.org/0000-0003-4305-2049>), Alexandra Maertens [ctb] (ORCID: <https://orcid.org/0000-0002-2077-2011>), JHU Discovery Award [fnd]

Maintainer Jason Laird <jasonlaird5353@gmail.com>

Contents

`.can_use_network` *Internal - should we allow network access now?*

Description

Controlled by the R option `tidyexposomics.allow_network` and `interactive()`.

Usage

```
.can_use_network()
```

Value

Logical scalar.

.get_ols_description *Internal - fetch OLS description for a term*

Description

Internal - fetch OLS description for a term

Usage

```
.get_ols_description(ontology_id, ontology_prefix)
```

Arguments

ontology_id	Term ID (e.g., "HP:0004322").
ontology_prefix	Prefix (e.g., "HP").

Value

A character scalar description or NA_character_.

.load_ontologies *Internal - load ontologies*

Description

Internal - load ontologies

Usage

```
.load_ontologies(use_demo = TRUE, ...)
```

Arguments

use_demo	Logical; if TRUE, load packaged demo objects hpo, ecto, chebi.
...	Optional named overrides: hpo, ecto, chebi.

Value

A list with elements hpo, ecto, chebi.

.onLoad *Internal - onLoad hook to register www assets*

Description

Registers `inst/app/www` as a Shiny resource path `"www"` if present.

Usage

```
.onLoad(libname, pkgname)
```

Value

Invisibly returns `NULL`

.ont_annot_build_server *Internal - builds ontology annotation server*

Description

Internal - builds ontology annotation server

Usage

```
.ont_annot_build_server(ontologies)
```

Arguments

`ontologies`	A list with `hpo`, `ecto`, `chebi` data.frames.

Value

A Shiny server function.

.ont_annot_build_ui *Internal - builds ontology annotation UI*

Description

Internal - builds ontology annotation UI

Usage

```
.ont_annot_build_ui()
```

Value

A `shiny.tag` UI definition.

.plot_exp_omic_category

Plot Significant Omics Associations per Exposure Category

Description

Internal helper that counts and plots the number of significant omics associations for each exposure category.

Usage

```
.plot_exp_omic_category(
  exposomicset,
  pval_col = "p_adjust",
  pval_thresh = 0.05
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing association results.
`pval_col`	Character. Name of the column used for p-value filtering. Defaults to `"p_adjust"`.
`pval_thresh`	Numeric. Significance threshold applied to `pval_col`. Rows with values below this threshold are retained. Defaults to `0.05`.

Value

A `ggplot` object.

.plot_exp_omic_exposure

Plot Significant Omics Associations per Exposure

Description

Internal helper that counts and plots the number of significant omics associations for each individual exposure.

Usage

```
.plot_exp_omic_exposure(
  exposomicset,
  pval_col = "p_adjust",
  pval_thresh = 0.05
)
```

Arguments

exposomicset	A MultiAssayExperiment object containing association results.
pval_col	Character. Name of the column used for p-value filtering. Defaults to "p_adjust".
pval_thresh	Numeric. Significance threshold applied to pval_col. Rows with values below this threshold are retained. Defaults to 0.05.

Value

A ggplot object.

.run_categorize_ontology

Internal - categorizes to ontology roots / depth

Description

Internal - categorizes to ontology roots / depth

Usage

```
.run_categorize_ontology(
  data,
  id_col,
  ontologyDF,
  root_level = 0,
  assign_label = TRUE
)
```

Arguments

data	Data frame with an ID column.
id_col	Column name containing ontology term IDs.
ontologyDF	Processed ontology data.frame (with list columns for relationships and depth).
root_level	Either a character vector of explicit root IDs, a numeric depth, or (default) top-level roots.
assign_label	If TRUE, return input with new columns; otherwise return assigned IDs.

Value

Data frame with root_id, root_label, category, category_source (if assign_label).

build_ont_annot_app *Build the Ontology Annotation Shiny app*

Description

Returns a Shiny app object for ontology annotation. This function does not launch the app. Please call `shiny::runApp()` yourself (see examples).

Usage

```
build_ont_annot_app(use_demo = TRUE, ...)
```

Arguments

`use_demo`	Logical, if TRUE, load packaged lightweight demo ontologies.
`...`	Optional named overrides passed as data.frames with columns `id`, `name`, `ancestors`, `parents`, `children`: `hpo`, `ecto`, `chebi`.

Value

A `shiny.appobj`.

Examples

```
if (interactive()) {
    app <- build_ont_annot_app()
    shiny::runApp(app)
}
```

create_exposomicset *Create an Exposomicset Object*

Description

Constructs a `MultiAssayExperiment` object from exposure data and optionally omics datasets, ensuring proper formatting and alignment of samples and features. For epidemiology-only workflows, omics data can be omitted.

Usage

```
create_exposomicset(codebook, exposure, omics = NULL, row_data = NULL)
```

Arguments

`codebook`	A data frame containing variable information metadata.
`exposure`	A data frame containing exposure data, with rows as samples and columns as variables.
`omics`	An optional list of matrices or a single matrix representing omics data. Each matrix should have samples as columns and features as rows. If `NULL`, creates an exposure-only exposomicset. Default is `NULL`.
`row_data`	An optional list of `DataFrame` objects providing feature metadata for each omics dataset. If `NULL`, row metadata is generated automatically. Default is `NULL`.

Details

The function validates inputs and creates a `MultiAssayExperiment` object. If omics data is provided, it converts matrices into `SummarizedExperiment` objects with proper sample alignment. If omics is `NULL`, the function creates an exposure-only object suitable for epidemiological analyses using `run_association()` with `source = "exposures"`.

Value

A `MultiAssayExperiment` object containing the formatted exposure and optionally omics datasets.

Examples

```
# Epi user workflow
# so no omics data
epi_data <- data.frame(
    pm25 = rnorm(10),
    outcome = rbinom(10, 1, 0.5),
    age = rnorm(10, 45, 10),
    row.names = paste0("subj_", 1:10)
)

codebook <- data.frame(
    variable = c("pm25", "outcome", "age"),
    category = c("exposure", "outcome", "covariate")
)

mae <- create_exposomicset(
    codebook = codebook,
    exposure = epi_data
)

# Multi-omics workflow
tmp <- make_example_data(n_samples = 10)

mae <- create_exposomicset(
    codebook = tmp$codebook,
    exposure = tmp$exposure,
    omics = tmp$omics,
    row_data = tmp$row_data
)
```

---

download_dataset *Download and cache a tidyexposomics dataset*

Description

Download and cache a tidyexposomics dataset

Usage

```
download_dataset(
 name = c("omics_list", "fdata", "meta", "annotated_cb", "expom_1", "chebi", "ecto",
    "hpo"),
  verbose = TRUE,
  validate = TRUE
)
```

Arguments

name	Dataset name: one of "omics_list", "fdata", "meta", "annotated_cb", "expom_1", "chebi", "ecto", "hpo".
verbose	Logical; print messages.
validate	Logical; validate MD5 checksum.

Value

A list or object loaded from the cached .RData.

---

extract_omics_exposure_df

Extract Merged Omics and Exposure Data Frame

Description

This function extracts and merges exposure variables from `colData` with selected features from omics datasets in a `MultiAssayExperiment` object. Optionally applies log2 transformation to omics data and restricts features based on a variable map.

Usage

```
extract_omics_exposure_df(exposomicset, variable_map = NULL, log2_trans = TRUE)
```

Arguments

exposomicset	A `MultiAssayExperiment` object containing omics and exposure data.
variable_map	A data frame with columns `"variable"` and `"exp_name"`, indicating which variables belong to each omics or exposure domain.
log2_trans	Logical; whether to log2-transform omics data. Default is `TRUE`.

Details

If `variable_map` is provided, it is used to select variables from both exposures and omics. If not provided, all numeric `colData` variables are used as exposures (excluding variables matching `^PC`), and all omics features are included.

Value

A data frame where rows correspond to samples, and columns contain exposure variables and log2-transformed omics features. Columns from different omics types are disambiguated using prefixes.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)
# export the omics exposure df
merged_df <- extract_omics_exposure_df(
    mae,
    log2_trans = TRUE
)
```

---

extract_results *Extract Results from* MultiAssayExperiment *Metadata*

Description

Retrieves a specific analysis result from the metadata slot of a MultiAssayExperiment object.

Usage

```
extract_results(
  exposomicset,
  result = c("codebook", "quality_control", "correlation", "association",
    "differential_analysis", "multiomics_integration", "network", "enrichment")
)
```

Arguments

exposomicset	A MultiAssayExperiment object.
result	A character string indicating which result to extract from metadata. Must be one of: "codebook", "quality_control", "correlation", "association", "mixture_analysis", "differential_analysis", "multiomics_integration", "network", "enrichment".

Value

The corresponding result object stored in metadata(exposomicset), or NULL if not present.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# extract results
res <- extract_results(
    exposomicset = mae,
    result = "codebook"
)
```

`extract_results_excel` *Export tidyexposomics Results to Excel*

Description

Exports selected results stored in a `MultiAssayExperiment` object created by the `tidyexposomics` pipeline to an Excel workbook. Users can select which result types to include, and optionally add placeholder sheets for missing data.

Usage

```
extract_results_excel(
  exposomicset,
  file = "tidyexposomics_results.xlsx",
  result_types = c("correlation", "association", "mixture_analysis",
    "differential_analysis", "multiomics_integration", "network", "enrichment",
    "exposure_summary", "pipeline"),
  include_empty_tabs = FALSE
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object with results stored in `@metadata`, typically created by the `tidyexposomics` pipeline.
`file`	Character. Path to the output Excel file.
`result_types`	Character vector specifying which result categories to export. Options include: • `"correlation"`: Correlation results. • `"association"`: Association results. • `"mixture_analysis"`: Mixture Analysis results. • `"differential_analysis"`: Differential abundance results, including sensitivity analysis if available. • `"multiomics_integration"`: Common top features contributing to latent factors from multi-omics integration. • `"network"`: Exposure impact metrics from network analyses. • `"enrichment"`: Enrichment results by omic and exposure category. • `"exposure_summary"`: Summary statistics for exposure variables. • `"pipeline"`: Overview of steps completed in the pipeline. Use `"all"` to export all of the above categories.
`include_empty_tabs`	Logical. If `TRUE`, adds placeholder sheets for any missing result types. Default is `FALSE`.

Value

An Excel file is written to the specified path. A message is printed with the file location.

Examples

```
# Create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# run correlation analysis
mae <- mae |>
    run_correlation(
        feature_type = "exposures",
        exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi")
    )

# file path of the output file
tmp <- tempfile(fileext = ".xlsx")

# extract the correlation results
extract_results_excel(
    exposomicset = mae,
    result_types = "correlation",
    file = tmp
)
```

---

extract_top_factor_features

Extract Top Contributing Features for Factors

Description

Identifies the most influential features for specified factors using multiomics integration results. Features are selected based on either a percentile cutoff or an absolute loading threshold.

Usage

```
extract_top_factor_features(
  exposomicset,
  factors = NULL,
  pval_col = "p_adjust",
  pval_thresh = 0.05,
  method = "percentile",
  percentile = 0.9,
  threshold = 0.3,
  action = "add"
)
```

Arguments

exposomicset	A `MultiAssayExperiment` object containing integration results.
factors	A character vector specifying the factors of interest. If `NULL`, factors are automatically selected from the association results using the `pval_col` and `pval_thresh` criteria.

pval_col	A string specifying the column name of the p-value or adjusted p-value used for factor selection if factors is NULL. Default is "p_adjust".
pval_thresh	A numeric value specifying the significance threshold for selecting factors from association results when factors is NULL. Default is 0.05.
method	A character string specifying the feature selection method ("percentile" or "threshold"). Default is "percentile".
percentile	A numeric value between 0 and 1 indicating the percentile threshold for feature selection when method = "percentile". Default is 0.9.
threshold	A numeric value specifying the absolute loading cutoff for feature selection when method = "threshold". Default is 0.3.
action	A character string indicating whether to return results ("get") or add them to metadata ("add"). Default is "add".

Details

The function extracts factor loadings from metadata(exposomicset), applies filtering based on the selected method, and identifies top contributing features for each specified factor.

If factors is not provided, the function will automatically select statistically significant factors from metadata(exposomicset)$association$assoc_factors$results_df using the specified pval_col and pval_thresh as criteria.

Features can be selected using:

- **Percentile-based filtering** (method = "percentile"): Selects features with absolute loadings above a specified percentile.
- **Threshold-based filtering** (method = "threshold"): Selects features with absolute loadings exceeding a fixed value.

Value

If action = "add", returns the modified exposomicset with selected features stored in metadata. If action = "get", returns a data frame containing:

feature	The selected feature contributing to the factor.
factor	The factor to which the feature contributes.
loading	The factor loading value of the feature.
exp_name	The experiment from which the feature originated.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# perform multiomics integration
mae <- run_multiomics_integration(
    mae,
    method = "DIABLO",
    outcome = "smoker",
    n_factors = 3
```

```
)

top_feats <- extract_top_factor_features(
    mae,
    factors = c("V1", "V2", "V3"),
    method = "percentile",
    percentile = 0.9,
    action = "get"
)
```

`filter_missing` *Filter Features and Variables with High Missingness*

Description

Removes exposure variables and omics features with missing values above a specified threshold. Generates missing data summaries and quality control (QC) plots.

Usage

```
filter_missing(exposomicset, na_thresh = 20, na_plot_thresh = 5)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing exposure and omics data.
`na_thresh`	A numeric value specifying the percentage of missing data allowed before a variable or feature is removed. Default is `20`.
`na_plot_thresh`	A numeric value specifying the minimum missing percentage for inclusion in QC plots. Default is `5`.

Details

The function assesses missingness in both `colData(exposomicset)` (exposure data) and `experiments(exposomicset)` (omics data).

- Exposure variables with more than `na_thresh`% missing values are removed.
- Omics features (rows in assay matrices) exceeding `na_thresh`% missing values are filtered.
- Missingness summaries and QC plots are generated using `naniar::gg_miss_var()` and stored in metadata.

Value

A `MultiAssayExperiment` object with filtered exposure variables and omics features. QC results, including missingness summaries and plots, are stored in `metadata(exposomicset)$na_qc`.

Examples

```
# Create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# Introduce some missingness
MultiAssayExperiment::colData(mae)$exposure_pm25[sample(1:20, 5)] <- NA

# Filter features and exposures with high missingness
mae_filtered <- filter_missing(
    exposomicset = mae,
    na_thresh = 20,
    na_plot_thresh = 5
)
```

`filter_non_normal` *Filter Non-Normal Exposure Variables*

Description

Removes exposure variables that deviate significantly from a normal distribution based on normality test results stored in metadata.

Usage

```
filter_non_normal(exposomicset, p_thresh = 0.05)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing exposure and omics data.
`p_thresh`	A numeric value specifying the p-value threshold for normality. Variables with `p.value` < `p_thresh` are removed. Default is `0.05`.

Details

The function identifies exposure variables that fail a normality test using `metadata(exposomicset)$transformation$r`

- Exposure variables with `p.value` < `p_thresh` are removed from `colData(exposomicset)`.
- The same filtering is applied to `colData` in each experiment within `experiments(exposomicset)`.

Value

A `MultiAssayExperiment` object with non-normal exposure variables removed.

Examples

```
# Create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# Test for normality
mae <- mae |>
    run_normality_check() |>
    transform_exposure(exposure_cols = c("age", "bmi", "exposure_pm25"))

# Remove non-normal variables
mae_filtered <- mae |>
    filter_non_normal()
```

`filter_omics` *Filter low-quality features in omics assays*

Description

This function applies variance- or expression-based filtering across one or more assays within a `MultiAssayExperiment` object. It is useful for removing low-quality or uninformative features before downstream analysis.

Usage

```
filter_omics(
  exposomicset,
  method = c("variance", "expression"),
  assays = NULL,
  assay_name = 1,
  min_var = 1e-05,
  min_value = 5,
  min_prop = 0.7,
  verbose = TRUE
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing omics assays.
`method`	Filtering method: either `"variance"` or `"expression"`.
`assays`	Character vector of assay names to filter. If `NULL`, all assays are filtered.
`assay_name`	Name or index of the assay within each `SummarizedExperiment` to use.
`min_var`	Minimum variance threshold (used if `method = "variance"`).
`min_value`	Minimum expression value (used if `method = "expression"`).
`min_prop`	Minimum proportion of samples exceeding `min_value` (used if `method = "expression"`).
`verbose`	Whether to print messages for each assay being filtered.

Value

A filtered `MultiAssayExperiment` object with updated assays and step record.

Examples

```
# Filter the proteomics assay by variance
filtered_mae <- filter_omics(
    exposomicset = make_example_data(return_mae = TRUE),
    method = c("variance"),
    assays = "proteomics",
    assay_name = 1,
    min_var = 0.01,
    verbose = TRUE
)
```

`filter_sample_outliers`

Filter Sample Outliers

Description

Removes sample outliers from a `MultiAssayExperiment` object based on PCA analysis.

Usage

```
filter_sample_outliers(exposomicset, outliers = NULL)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing omics and exposure data.
`outliers`	An optional character vector specifying sample names to be removed. If `NULL`, the function uses outliers identified in `metadata(exposomicset)$pca$outliers`. Default is `NULL`.

Details

The function checks for the presence of PCA results in `metadata(exposomicset)`. If `outliers` is not provided, it retrieves precomputed outliers from `metadata(exposomicset)$pca$outliers`. The identified samples are removed from the dataset.

Value

A `MultiAssayExperiment` object with the specified outliers removed.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# run PCA
mae <- mae |>
    run_pca()

# filter outliers if present
mae <- mae |>
    filter_sample_outliers()
```

---

load_annotation_data *Load Ontology Data*

Description

Downloads and loads chebi, ecto, and hpo.

Usage

```
load_annotation_data(
  to_load = c("all", "chebi", "ecto", "hpo"),
  verbose = TRUE,
  validate = TRUE
)
```

Arguments

to_load	Character vector indicating which ontology to load.
verbose	Logical; print messages.
validate	Logical; validate MD5 checksum.

Value

A named list of ontology objects.

Examples

```
## Not run:
# load single ontology
onts <- load_annotation_data(to_load = "ecto")

# load all ontologies
onts <- load_annotation_data(to_load = "all")

## End(Not run)
```

`make_example_data` *Generate Example Data for Testing*

Description

This helper function generates a reproducible dummy dataset containing exposures, mRNA data, and proteomics data. It can optionally return the data as a `MultiAssayExperiment` using `create_exposomicset`.

Usage

```
make_example_data(
  n_samples = 12,
  n_proteins = 80,
  use_batch = FALSE,
  return_mae = FALSE
)
```

Arguments

`n_samples`	Integer. Number of samples to simulate (default: 12).
`n_proteins`	Integer. Number of proteins to simulate (default: 80).
`use_batch`	Logical. If `TRUE`, include a "batch" variable in the exposure data (default: FALSE).
`return_mae`	Logical. If `TRUE`, return a `MultiAssayExperiment` created using `create_exposomicset()` (default: FALSE).

Value

Either:

- A named list containing `codebook`, `exposure`, `omics`, and `row_data`, if `return_mae = FALSE`.
- A `MultiAssayExperiment`, if `return_mae = TRUE`.

Examples

```
# Return as a list
dummy <- make_example_data()

# Return as a MultiAssayExperiment
mae <- make_example_data(return_mae = TRUE)
```

`pivot_exp` *Pivot a selected omics dataset from a MultiAssayExperiment into tidy-bulk format*

Description

Extracts a specified omics dataset from a `MultiAssayExperiment`, optionally filters by feature (row) names, and returns a tidy tibble. The output includes assay values along with sample metadata and feature metadata.

Usage

```
pivot_exp(exposomicset, exp_name, features = NULL)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing one or more omics assays.
`exp_name`	A character string. The name of the omics dataset to extract (e.g., "Proteomics").
`features`	Optional character vector of row (feature) names to retain. If `NULL`, all features are included.

Value

A tibble in tidy format with one row per feature/sample pair, including all metadata and a new column `exp_name` indicating the assay source. Assay values are provided in separate columns named after the assay slot(s).

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# pivot experiment
exp_data <- pivot_exp(
    exposomicset = mae,
    exp_name = "mRNA",
    features = c("feat_42")
)
```

pivot_feature *Extract Feature Metadata from a MultiAssayExperiment*

Description

Extracts feature-level metadata across all assays in a `MultiAssayExperiment` and returns a combined tibble.

Usage

```
pivot_feature(exposomicset)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object.

Details

This function:

- Iterates over all assays in the `MultiAssayExperiment`.
- Updates each assay's sample metadata (`colData`) using `.update_assay_colData()`.
- Extracts feature-level metadata using `pivot_transcript()` from `tidybulk`.
- Combines results across assays into a single tibble, adding a `.exp_name` column.

Value

A tibble with feature metadata from all assays, with an added `.exp_name` column.

Examples

```
#' # create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# pivot experiment
feature_data <- mae |>
    pivot_feature()
```

pivot_sample *Extract Sample Metadata from MultiAssayExperiment or SummarizedExperiment*

Description

Extracts and formats sample-level metadata (`colData`) from a `MultiAssayExperiment` or `SummarizedExperiment` object.

Usage

```
pivot_sample(x, ...)
```

Arguments

`x`	A `MultiAssayExperiment` or `SummarizedExperiment` object.
`...`	Additional arguments passed to `pivot_sample()` from `tidybulk` for `SummarizedExperiment` objects.

Details

This function:

- Extracts **sample metadata** from `MultiAssayExperiment` using `colData()`, converting it to a tibble.
- Calls `pivot_sample()` from `tidybulk` when applied to a `SummarizedExperiment` object.
- **Error Handling**: Returns an error if `x` is not a `MultiAssayExperiment` or `SummarizedExperiment`.

Value

A tibble containing sample metadata with an added `.sample` column.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

sample_data <- mae |>
    pivot_sample()
```

plot_association *Plot Association Results (Unified Forest Plot)*

Description

Generates a forest plot for association results from any source stored in the metadata of a `MultiAssayExperiment` object. Supports faceting and visual augmentation with R^2 tiles when available.

Usage

```
plot_association(
  exposomicset,
  source = c("omics", "exposures", "factors", "go_pcs"),
  terms = NULL,
  filter_col = "p.value",
  filter_thresh = 0.05,
  direction_filter = "all",
  add_r2_tile = TRUE,
  r2_col = "adj_r2",
  facet = FALSE,
  nrow = 1,
  subtitle = NULL
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing association results in metadata.
`source`	Character string indicating the association source. One of `"omics"`, `"exposures"`, `"factors"`, or `"go_pcs"`.
`terms`	Optional character vector of term names to subset the plot to. Default is `NULL` (include all).
`filter_col`	Column used to assess statistical significance (default: `"p.value"`).
`filter_thresh`	Numeric threshold applied to `filter_col` (default: `0.05`).
`direction_filter`	Direction of associations to retain. One of `"all"` (default), `"up"`, or `"down"`.
`add_r2_tile`	Logical; if `TRUE`, includes a tile plot for `r2_col` (default: `TRUE`).
`r2_col`	Column used for coloring the tile plot (default: `"adj_r2"`).
`facet`	Logical; if `TRUE` and `source == "go_pcs"`, apply nested faceting by experiment and GO cluster (default: `FALSE`).
`nrow`	Integer; number of rows for facet layout if enabled (default: `1`).
`subtitle`	Optional subtitle for the plot. If `NULL`, automatically generated from covariates used in the model.

Details

This function visualizes effect size estimates and confidence intervals from association analyses. It allows filtering by direction (`"up"` for positive, `"down"` for negative) and by significance. For `source = "go_pcs"`, it supports special formatting by splitting term labels into nested facets.

The R^2 tile (if enabled) adds a side heatmap indicating model fit for each association. This can be useful for model diagnostics.

Value

A ggplot2 object: either a single forest plot or a composite plot with an R^2 tile strip.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# run association tests
mae <- mae |>
    run_association(
        source = "exposures",
        feature_set = c("exposure_pm25", "exposure_no2"),
        outcome = "smoker",
        covariates = c("age"),
        family = "binomial"
    )

assoc_plot <- mae |>
    plot_association(
        source = "exposures"
    )
```

---

plot_circos_correlation

Plot Circular Network of Exposure Relationships

Description

Generates a circular network plot to visualize relationships between exposures, either based on correlation ("exposures") or shared features ("degs", "factors").

Usage

```
plot_circos_correlation(
  exposomicset,
  feature_type = c("degs", "omics", "factors", "factor_features", "exposures", "pcs"),
  exposure_cols = NULL,
  corr_threshold = NULL,
  shared_cutoff = 10,
  annotation_colors = NULL,
  low = "#006666",
  mid = "white",
  high = "#8E0152",
  midpoint = NULL
)
```

Arguments

exposomicset	A MultiAssayExperiment object.
feature_type	One of "exposures", "degs", or "factors".
exposure_cols	Character vector of exposures to include (only for "exposures").
corr_threshold	Minimum |correlation| (only for "exposures").
shared_cutoff	Minimum number of shared features (only for "degs" or "factors"). Default = 10.
annotation_colors	Optional named vector of colors for categories.
low	low value color for edges.
mid	middle value color for edges.
high	high value color for edges.
midpoint	Midpoint for edge color gradient. Defaults to 0 (for correlations) or mean shared features.

Value

A ggplot object (ggraph circular plot).

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# run correlation analysis
mae <- mae |>
    run_correlation(
        feature_type = "exposures",
        exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi")
    )

# create the circos plot
circos_plot <- mae |>
    plot_circos_correlation(
        feature_type = "exposures"
    )
```

---

plot_correlation_summary

Plot Correlation Summary from Exposure-Feature Correlations

Description

Generates a bar plot summary of exposure-feature correlations using customizable modes.

Usage

```
plot_correlation_summary(
  exposomicset,
 feature_type = c("degs", "omics", "factors", "factor_features", "exposures", "pcs"),
 mode = c("top_exposures", "top_features", "exposure_category", "assay", "summary"),
  top_n = 15
)
```

Arguments

exposomicset	A MultiAssayExperiment object with correlation results in metadata.
feature_type	One of "degs", "factors", "omics", or "exposures".
mode	One of: • "top_exposures": Top exposures by assay • "top_features": Top features by exposure category • "exposure_category": Total associations by exposure category • "assay": Total associations by omics assay • "summary": Patchwork layout combining all
top_n	Number of top exposures or features to display (for top modes). Default is 15.

Value

A ggplot2 object or a patchwork object (if mode = "summary").

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# correlate with exposures
mae <- mae |>
    run_correlation(
        feature_type = "omics",
        variable_map = mae |>
            pivot_feature() |>
            dplyr::select(
                variable = .feature,
                exp_name = .exp_name
            ),
        exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi")
    )

# create the correlation summary plot
cor_summary_plot <- mae |>
    plot_correlation_summary(
        feature_type = "omics",
        mode = "summary"
    )
```

plot_correlation_tile *Plot Correlation Tilemap*

Description

Visualizes a correlation matrix as a heatmap tile plot using correlation results stored in the metadata of a `MultiAssayExperiment` object. When `feature_type = "pcs"`, the function forces PCs to appear on the x-axis and exposures on the y-axis, and it adds a barplot showing how many PCs are significantly associated with each exposure. It also suppresses nonsignificant tiles based on a specified p-value threshold.

Usage

```
plot_correlation_tile(
  exposomicset,
  feature_type = c("pcs", "degs", "omics", "factors", "factor_features", "exposures"),
  pval_cutoff = 0.05,
  na_color = "grey100",
  fill_limits = c(-1, 1),
  midpoint = 0
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing correlation results in metadata.
`feature_type`	Type of correlation results to plot. One of `"pcs"`, `"degs"`, `"omics"`, `"factors"`, `"factor_features"`, or `"exposures"`. Must match the key used in `metadata(exposomicset)$corr`
`pval_cutoff`	Numeric p-value cutoff below which correlations are displayed. Nonsignificant tiles are rendered in the `na_color`. Default is `0.05`.
`na_color`	Color used to represent nonsignificant or missing correlations. Default is `"grey100"`.
`fill_limits`	Numeric vector of length 2 defining the scale limits for correlation values. Default is `c(-1, 1)`.
`midpoint`	Numeric value for centering the fill gradient. Default is `0`.

Value

A `ggplot2` tile plot (or a combined tile + barplot if `feature_type = "pcs"`).

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# run pca
mae <- mae |>
    run_pca()

# correlate with exposures
```

```
mae <- mae |>
    run_correlation(
        feature_type = "pcs",
        exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi")
    )

# make the correlation tile plot
cor_tile_p <- mae |>
    plot_correlation_tile(
        feature_type = "pcs"
    )
```

---

plot_enrichment *Plot Enrichment Results from exposomicset*

Description

Visualize enrichment results stored in a `MultiAssayExperiment` object. Supports dotplots, heatmaps, cnetplots, networks, and multi-panel summary plots.

Usage

```
plot_enrichment(
  exposomicset,
  feature_type = c("degs", "degs_robust", "omics", "factor_features", "degs_cor",
    "omics_cor", "factor_features_cor"),
  plot_type = c("dotplot", "cnet", "network", "heatmap", "summary"),
  top_n = 5,
  n_per_group = 5,
  add_top_genes = TRUE,
  top_n_genes = 5,
  heatmap_fill = TRUE,
  logfc_thresh = log2(1),
  pval_col = "P.Value",
  pval_thresh = 0.05,
  score_metric = "stability_score",
  score_thresh = NULL,
  overlap_thresh = 0.2,
  node_radius = 0.2,
  pie_colors = NULL,
  label_top_n = NULL,
  label_colour = "black",
  net_facet_by = NULL,
  max_terms = 30,
  node_size = 1,
  term_node_correction = 0.2,
  gene_node_correction = 3,
  go_groups = NULL,
  layout_algo = "fr",
  edge_alpha = 0.3,
  label_size = 3,
```

```
  feature_col = "feature",
  logfc_col = "logFC"
)
```

Arguments

exposomicset	A MultiAssayExperiment object with enrichment results added via run_enrichment().
feature_type	Character; one of "degs", "degs_robust", "omics", "factor_features", "degs_cor", "omics_cor", or "factor_features_cor". Defines which enrichment results to use.
plot_type	Type of plot to generate. One of "dotplot", "cnet", "network", "heatmap", or "summary".
top_n	Integer; number of top go_groups to include (used in "dotplot"). Default is 5.
n_per_group	Integer; number of terms per group to plot (used in "dotplot"). Default is 5.
add_top_genes	Logical; if TRUE, appends top shared genes to dotplot facets. Default is TRUE.
top_n_genes	Integer; number of top genes to show in each group (used in "dotplot"). Default is 5.
heatmap_fill	Logical; whether to fill tiles by logFC in the heatmap. Default is TRUE.
logfc_thresh	Numeric; log2 fold change threshold for filtering (heatmap only). Default is log2(1).
pval_col	Column name of the p-value used for filtering in "degs" heatmap. Default is "P.Value".
pval_thresh	Threshold for pval_col (heatmap only). Default is 0.05.
score_metric	Column for stability score (used in "degs_robust" heatmap). Default is "stability_score".
score_thresh	Numeric; threshold for score_metric (heatmap only). Default is NULL.
overlap_thresh	Numeric; Jaccard threshold for edges in the network plot. Default is 0.2.
node_radius	Numeric; node size in network plot. Default is 0.2.
pie_colors	Optional named vector of colors for pie charts (network and cnet).
label_top_n	Integer; number of top nodes to label in network. Default is NULL.
label_colour	Color of node labels in network. Default is "black".
net_facet_by	Column used to facet the network plot (e.g., "category"). Default is NULL.
max_terms	Integer; max number of terms to include in the cnet plot. Default is 30.
node_size	Numeric; base node size for cnet plot. Default is 1.
term_node_correction	Scaling factor for term nodes in cnet plot. Default is 0.2.
gene_node_correction	Scaling factor for gene nodes in cnet plot. Default is 3.
go_groups	Optional character vector of GO group names to subset enrichment results (all plots).
layout_algo	Graph layout algorithm to use in "network" and "cnet" plots. Default is "fr".
edge_alpha	Transparency of network/cnet plot edges. Default is 0.3.
label_size	Font size for labels in network and cnet plots. Default is 3.
feature_col	Column name used to join gene-level metadata. Default is "feature".
logfc_col	Column name used for log2 fold change values. Default is "logFC".

Details

This function visualizes results from `run_enrichment()` using one of several plot types:

- `"dotplot"`: Enrichment terms grouped by GO group, colored by significance.
- `"heatmap"`: Term - gene matrix with optional logFC fill and shared gene highlighting.
- `"network"`: Graph of term overlap based on shared genes, faceted by metadata if desired.
- `"cnet"`: Gene - term bipartite graph with gene logFC values and term pie slices.
- `"summary"`: Multi-panel figure with GO group ridgeplots, gene counts, and Venn diagram.

Value

A `ggplot` or `patchwork` object corresponding to the requested plot type.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 30,
    return_mae = TRUE
)

# perform differential abundance analysis
mae <- run_differential_abundance(
    exposomicset = mae,
    formula = ~ smoker + sex,
    abundance_col = "counts",
    method = "limma_voom",
    action = "add"
)

# perform enrichment analysis
mae <- run_enrichment(
    exposomicset = mae,
    feature_type = "degs",
    feature_col = "symbol",
    species = "goa_human",
    deg_logfc_threshold = log2(1),
    deg_pval_col = "P.Value",
    deg_pval_threshold = 0.2,
    action = "add"
)

# create an enrichment plot
enr_plot <- plot_enrichment(
    exposomicset = mae,
    feature_type = "degs",
    plot_type = "network"
)
```

plot_exposures *Plot Exposure Distributions by Category or Group*

Description

Visualizes exposure variable distributions using **boxplots** or **ridge plots**.

Usage

```
plot_exposures(
  exposomicset,
  exposure_cat = "all",
  exposure_cols = NULL,
  group_by = NULL,
  plot_type = "boxplot",
  alpha = 0.3,
  panel_sizes = rep(1, 100),
  title = "Exposure Levels by Category",
  xlab = "",
  ylab = "",
  facet_cols = NULL,
  group_cols = NULL,
  box_width = 0.1,
  fill_lab = ""
)
```

Arguments

exposomicset	A MultiAssayExperiment object containing exposure data.
exposure_cat	A character string or vector specifying exposure category names (from codebook$category) to include. Use "all" to include all exposures.
exposure_cols	Optional character vector specifying exact exposure variables to plot.
group_by	A string specifying the column in colData(exposomicset) used to fill the plot (e.g., "sex"). Defaults to NULL, in which case exposures are colored by category.
plot_type	Type of plot: "boxplot" (default) or "ridge".
alpha	Transparency level for background facet color strips. Default is 0.5.
panel_sizes	A numeric vector passed to ggh4x::force_panelsizes() for controlling facet widths or heights.
title	Plot title. Default is "Exposure Levels by Category".
xlab	X-axis label. Default is an empty string.
ylab	Y-axis label. Default is an empty string.
facet_cols	Optional vector of colors to use as background for facet categories. If NULL, a default palette is used.
group_cols	Optional named vector of colors for group_by levels. If NULL, a default palette is used.
box_width	A numeric value specifying the width of the boxplots. Only used when plot_type = "boxplot". Default is 0.1.

`fill_lab` Legend title for the fill aesthetic (e.g., `"Sex"` or `"Exposure Group"`). Default is `""`.

Details

This function:

- Filters exposure data based on category or selected columns.
- Merges variable metadata from `metadata(exposomicset)$codebook`.
- Supports either **boxplot** (vertical distributions per variable) or **ridgeplot** (horizontal density plots per variable).
- If `group_by` is specified, that variable defines the plot fill color; otherwise, the fill is based on exposure `category`.
- Facets by `category` using `ggh4x::facet_grid2()` with color-coded strip backgrounds.
- The `box_width` argument controls the width of the boxplots when `plot_type = "boxplot"`.

Value

A `ggplot2` object showing exposure distributions, optionally grouped.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# plot exposure data
exposure_plot <- mae |>
    plot_exposures(
        exposure_cols = c("exposure_pm25", "exposure_no2"),
        box_width = 0.2
    )
```

---

plot_exposure_impact *Plot Exposure Impact on Network Centrality Metrics*

Description

Visualizes the impact of exposures on network centrality measures of associated features (e.g., genes or latent factors) as a heatmap. Each exposure is scored by four centrality metrics, scaled within metric, and grouped by exposure category.

Usage

```
plot_exposure_impact(
  exposomicset,
  feature_type = c("degs", "omics", "factors"),
  min_per_group = 5,
  facet_cols = NULL,
  bar_cols = NULL,
  alpha = 0.3,
  ncol = 2,
  nrow = 1,
  heights = c(1, 1),
  widths = c(2, 1)
)
```

Arguments

exposomicset	A MultiAssayExperiment object with results from run_exposure_impact().
feature_type	Character string specifying the feature type. One of "degs", "omics", or "factors".
min_per_group	Minimum number of features per exposure for inclusion (not currently used). Default is 5.
facet_cols	Optional named vector of colors for exposure categories.
bar_cols	Optional vector of colors for bar plots (if enabled).
alpha	Transparency level for category strips (if enabled). Default is 0.3.
ncol, nrow	Layout for optional patchwork combination (currently unused). Default: ncol = 2, nrow = 1.
heights	Relative heights and widths for combined plots (currently unused). Default: c(1,1).
widths	Relative widths for combined plots (currently unused). Default: c(2,1).

Details

This function uses the output of run_exposure_impact() to calculate and visualize the mean centrality values for each exposure across its associated features. The following network centrality metrics are shown:

- Degree centrality
- Eigenvector centrality
- Closeness centrality
- Betweenness centrality

All values are scaled within metric across exposures. A side bar indicates the category of each exposure.

Value

A ggplot/patchwork object showing a heatmap of scaled network centrality scores per exposure, annotated by category.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# perform correlation analyses
# correlate with exposures
mae <- mae |>
    run_correlation(
        feature_type = "omics",
        variable_map = mae |>
            pivot_feature() |>
            dplyr::select(
                variable = .feature,
                exp_name = .exp_name
            ),
        exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi")
    ) |>
    run_correlation(
        feature_type = "omics",
        variable_map = mae |>
            pivot_feature() |>
            dplyr::select(
                variable = .feature,
                exp_name = .exp_name
            ),
        feature_cors = TRUE,
        exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi")
    )

# create the networks
mae <- mae |>
    run_create_network(
        feature_type = "omics_feature_cor",
        action = "add"
    ) |>
    run_create_network(
        feature_type = "omics",
        action = "add"
    )

# perform impact analysis
mae <- mae |>
    run_exposure_impact(
        feature_type = "omics"
    )

# create the exposure impact plot
exposure_impact_p <- mae |>
    plot_exposure_impact(
        feature_type = "omics"
    )
```

`plot_exposure_omics_association` *Plot Exposure-Omics Associations*

Description

Plots the number of significant exposure-omics associations, grouped either by exposure or the exposure category.

Usage

```
plot_exposure_omics_association(
  exposomicset,
  plot_type = c("exposures", "category"),
  pval_col = "p_adjust",
  pval_thresh = 0.05
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing association results.
`plot_type`	Character. One of `"exposures"` or `"category"`. Controls whether associations are summarized per exposure or per exposure category. Defaults to `"exposures"`.
`pval_col`	Character. Name of the column used for p-value filtering. Defaults to `"p_adjust"`.
`pval_thresh`	Numeric. Significance threshold applied to `pval_col`. Rows with values below this threshold are retained. Defaults to `0.05`.

Value

A `ggplot` object.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# run exposure-omics association
mae <- mae |>
    run_exposure_omics_association(
        exposures = c("exposure_pm25", "exposure_no2"),
        covariates = c("age", "sex")
    )

plot_exposure_omics_association(
    exposomicset = mae,
    plot_type = "exposures"
)
```

plot_factor_summary *Plot Summary of Factor Contributions from Multi-Omics Integration*

Description

Generates a summary plot of factor contributions from multi-omics integration results stored in a `MultiAssayExperiment` object.

Usage

```
plot_factor_summary(
  exposomicset,
  low = "#006666",
  mid = "white",
  high = "#8E0152",
  midpoint = 0.5
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing integration results in `metadata(exposomicset)$multi`
`low`	Color for low values in the fill gradient. Default is `"#006666"`.
`mid`	Color for midpoint in the fill gradient. Default is `"white"`.
`high`	Color for high values in the fill gradient. Default is `"#8E0152"`.
`midpoint`	Midpoint value for the gradient color scale. Default is `0.5`.

Details

This function visualizes factor contributions based on the integration method:

- **MOFA**: Variance explained per factor and view.
- **MCIA**: Block score weights per omic.
- **DIABLO**: Mean absolute sample score per omic and factor (from block-specific variates).
- **RGCCA**: Mean absolute sample score per omic and factor (from aligned block scores).

The color gradient can be customized using the `low`, `mid`, `high`, and `midpoint` parameters.

Value

A `ggplot` object showing factor contributions based on the integration method.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

mae <- run_multiomics_integration(
    mae,
```

```
    method = "DIABLO",
    outcome = "smoker",
    n_factors = 3
)

factor_sum_plot <- mae |>
    plot_factor_summary()
```

---

plot_manhattan *Plot a Manhattan-style ExWAS summary across omics categories*

Description

This function generates a multi-faceted Manhattan plot from the results of `associate_all_outcome()`, visualizing the significance of associations across omics features, grouped by category. Significant features can be highlighted and labeled, and strip backgrounds can be colored per facet.

Usage

```
plot_manhattan(
  exposomicset,
  pval_thresh = 0.05,
  feature_col = "term",
  alpha = 0.5,
  min_per_cat = 1,
  vars_to_label = NULL,
  sig_color = "magenta2",
  non_sig_cols = c("grey25", "grey75"),
  pval_thresh_line_col = "grey25",
  panel_sizes = c(1, 1, 1, 1, 1),
  linetype = "dashed",
  facet_cols = NULL,
  label_size = 3.5,
  facet_angle = 90,
  facet_text_face = "bold.italic"
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object that has already been processed by `associate_all_outcome()`.
`pval_thresh`	Numeric threshold for significance (default = 0.05).
`feature_col`	A character string indicating the column name to use for feature labeling and highlighting (e.g., `"term"` or `"feature"`). Default is `"term"`.
`alpha`	Transparency applied to facet strip colors (default = 0.5).
`min_per_cat`	Minimum number of features per category to be shown (default = 1).
`vars_to_label`	Optional character vector of variable names to label explicitly, matched against the `feature_col` column.
`sig_color`	Color used for significant points (default = `"magenta2"`).

non_sig_cols Character vector of alternating colors for non-significant points (default = c("grey25", "grey75")).

pval_thresh_line_col Color of the horizontal significance threshold line (default = "grey25").

panel_sizes Numeric vector passed to ggh4x::force_panelsizes() to control panel widths (default = c(1,1,1,1,1)).

linetype Line type for the horizontal threshold (default = "dashed").

facet_cols Optional vector of colors to use for facet strip backgrounds.

label_size Numeric size of the feature label text (default = 3.5).

facet_angle Angle (in degrees) for strip text rotation (default = 90).

facet_text_face Font face for facet strip labels (default = "bold.italic").

Details

- This function expects associate_all_outcome() to have been run first.
- Facets represent omics categories, and points represent features.
- Points below the significance threshold are colored using non_sig_cols, while significant ones are colored with sig_color and optionally labeled.
- Uses ggrepel to avoid overlapping labels and ggh4x for enhanced faceting.
- The feature_col argument allows customization of which column is used to label or identify features, enabling compatibility with different result formats.

Value

A ggplot object showing the Manhattan-style faceted plot.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# run association tests
mae <- mae |>
    run_association(
        source = "omics",
        top_n = 20,
        feature_set = c("exposure_pm25", "exposure_no2"),
        outcome = "smoker",
        covariates = c("age"),
        family = "binomial"
    )

# create the manhattan plot
manhattan_p <- mae |>
    plot_manhattan(feature_col = "term")
```

plot_missing *Plot Missing Data Across Exposure and Omic Layers*

Description

Visualizes missing data patterns in a `MultiAssayExperiment` object using summary bar plots or feature-level lollipop plots.

Usage

```
plot_missing(
  exposomicset,
  threshold = 5,
  plot_type = c("summary", "lollipop"),
  layers = NULL
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing exposure and omics assays. Missing data is inferred directly from the assays.
`threshold`	Numeric. The percentage threshold (0-100) above which features are counted as missing in the summary plot. Default is `5`.
`plot_type`	Character. Type of plot to generate. Either `"summary"` for a bar plot showing number of features above the missing threshold, or `"lollipop"` for a per-feature lollipop plot with layer annotations. Default is `"summary"`.
`layers`	Optional character vector. If specified, filters the plot to include only selected layers (e.g., `"Exposure"`, `"Transcriptome"`).

Details

The function calculates missing data per feature (or variable) across all assays (including exposure variables) and generates:

- **Summary plot** (`plot_type = "summary`): A bar plot showing the number of variables in each assay exceeding the specified missingness threshold.
- **Lollipop plot** (`plot_type = "lollipop`): A feature-level plot where each feature's percent missingness is shown, along with a color-coded tile on the side indicating its layer of origin.

The tile colors in the lollipop plot match the experiment colors used in other visualizations (e.g., via `scale_color_tidy_exp()`).

Value

A `ggplot` or `patchwork` object depending on the selected `plot_type`.

Examples

```
#' # Create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# Introduce some missingness
MultiAssayExperiment::colData(mae)$exposure_pm25[sample(1:20, 5)] <- NA

# Summary bar plot of missing data
summary_p <- plot_missing(
    mae,
    threshold = 10,
    plot_type = "summary"
)

# Lollipop plot for all features with any missingness
lollipop_p <- plot_missing(
    mae,
    plot_type = "lollipop"
)
```

`plot_network` *Plot Network Graph of Features or Exposures*

Description

Visualizes network structures created by `run_create_network()` from the metadata of a `MultiAssayExperiment` object. Nodes can represent features (e.g., genes or factors) or exposures, and edges represent correlations or shared connections.

Usage

```
plot_network(
  exposomicset,
  network = c("degs", "omics", "factors", "factor_features", "exposures",
    "degs_feature_cor", "omics_feature_cor", "factor_features_feature_cor"),
  include_stats = TRUE,
  nodes_to_include = NULL,
  centrality_thresh = NULL,
  top_n_nodes = NULL,
  cor_thresh = NULL,
  label = FALSE,
  label_top_n = 5,
  nodes_to_label = NULL,
  facet_var = NULL,
  foreground = "steelblue",
  fg_text_colour = "grey25",
  node_colors = NULL,
  node_color_var = NULL,
```

```
  alpha = 0.5,
  size_lab = "Centrality",
  color_lab = "Group"
)
```

Arguments

exposomicset	A MultiAssayExperiment object containing network results in metadata.
network	Character string specifying the network type. One of "degs", "omics", "factors", "factor_features", "exposures", "degs_feature_cor", "omics_feature_cor", "factor_features_feature_cor".
include_stats	Logical; if TRUE, include edge weights and node centrality metrics in the plot aesthetics. Default is TRUE.
nodes_to_include	Optional character vector of node names to include (subset of name).
centrality_thresh	Optional numeric threshold to filter nodes by centrality degree.
top_n_nodes	Optional integer to keep only the top N nodes by centrality.
cor_thresh	Optional numeric threshold to filter edges by minimum absolute correlation.
label	Logical; whether to label nodes. If TRUE, top nodes will be labeled.
label_top_n	Integer; number of top-centrality nodes to label if label = TRUE. Default is 5.
nodes_to_label	Optional character vector of specific nodes to label.
facet_var	Optional node metadata column to facet the network layout by.
foreground	Color for node outlines and edge borders. Default is "steelblue".
fg_text_colour	Color of node label text. Default is "grey25".
node_colors	Optional named vector of colors for node groups.
node_color_var	Optional node attribute used for node color mapping.
alpha	Alpha transparency for nodes and edges. Default is 0.5.
size_lab	Legend title for node size (typically centrality). Default is "Centrality".
color_lab	Legend title for node color group. Default is "Group".

Details

This function retrieves the stored graph object and optionally filters or labels nodes based on: centrality, correlation, user input, or group-specific attributes. It supports layout faceting, custom color mappings, and highlights highly central nodes.

Large graphs (> 500 nodes) will prompt the user before plotting.

Value

A ggraph plot object.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# perform correlation analyses
# correlate with exposures
mae <- mae |>
    run_correlation(
        feature_type = "omics",
        variable_map = mae |>
            pivot_feature() |>
            dplyr::select(
                variable = .feature,
                exp_name = .exp_name
            ),
        exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi")
    )

# create the networks
mae <- mae |>
    run_create_network(
        feature_type = "omics",
        action = "add"
    )

# plot the network
network_p <- mae |>
    plot_network(
        network = "omics"
    )
```

`plot_normality_summary` *Plot Normality Summary of Exposure Variables*

Description

Generates a bar plot summarizing the number of exposure variables that pass or fail normality tests (e.g., Shapiro-Wilk) before or after transformation.

Usage

```
plot_normality_summary(exposomicset, transformed = FALSE)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object with quality control metadata.
`transformed`	Logical; if `TRUE`, use results after transformation. Default is `FALSE`.

Details

This function assumes that `run_normality_check()` has been executed and that the results are stored in `metadata(exposomicset)$quality_control$normality`. If `transformed = TRUE`, the function will instead plot the transformation summary stored in `metadata(exposomicset)$quality_control$transfo` which is populated by `transform_exposure()`.

The plot includes both bar heights and overlaid line segments to reinforce the counts.

Value

A `ggplot` object summarizing the number of exposures classified as normal or not normal.

Examples

```
# Create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# Test for normality
mae <- mae |>
    run_normality_check() |>
    transform_exposure(exposure_cols = c("age", "bmi", "exposure_pm25"))

# plot the normality summary
norm_p <- mae |>
    plot_normality_summary()
```

---

plot_pca *Plot PCA Results for Features and Samples*

Description

Generates PCA plots for both feature space and sample space, including scatter plots and scree plots.

Usage

```
plot_pca(
  exposomicset,
  feature_col = "#00a9b2",
  sample_col = "#8a4f77",
  sample_outlier_col = "firebrick"
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing PCA results in `metadata(exposomicset)$pca`.
`feature_col`	A character string specifying the color for the feature scree plot. Default is `"#00a9b2"`.

sample_col A character string specifying the color for the sample scree plot. Default is "#8a4f77".

sample_outlier_col A character string specifying the color for sample outlier labels. Default is "firebrick".

Details

This function creates four PCA visualizations:

- **Feature Space PCA Plot**: Colored by category (e.g., omics, exposure).
- **Feature Scree Plot**: Displays the variance explained by each principal component.
- **Sample Space PCA Plot**: Highlights outlier samples.
- **Sample Scree Plot**: Displays variance explained in the sample PCA.

Outliers are labeled based on `metadata(exposomicset)$pca$outliers`.

Value

A combined ggplot object containing the four PCA plots.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# run pca
mae <- mae |>
    run_pca()

# create the pca plot
pca_p <- mae |>
    plot_pca()
```

---

plot_sample_clusters *Plot Sample Clusters*

Description

Generates a heatmap of sample clustering results and summarizes sample group assignments.

Usage

```
plot_sample_clusters(exposomicset, exposure_cols = NULL)
```

Arguments

exposomicset A MultiAssayExperiment object containing sample clustering results in metadata(exposomicset)

exposure_cols A character vector specifying columns from colData to include in the summary. Default is NULL, which includes all available columns.

Details

This function:

- Extracts sample cluster assignments from `metadata(exposomicset)$sample_clustering`.
- Merges cluster labels with `colData(exposomicset)`.
- Plots the heatmap stored in `metadata(exposomicset)$sample_clustering$heatmap`.

Value

A `ComplexHeatmap` plot displaying sample clustering results.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 30,
    return_mae = TRUE
)

# determine sample clusters
mae <- run_cluster_samples(
    exposomicset = mae,
    exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi"),
    clustering_approach = "diana"
)

# plot sample clusters
sample_cluster_p <- mae |>
    plot_sample_clusters(
        exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi")
    )
```

---

plot_sensitivity_summary

Plot Sensitivity Analysis Summary

Description

Generates a ridge plot and bar chart summarizing feature stability scores across assays.

Usage

```
plot_sensitivity_summary(
  exposomicset,
  stability_score_thresh = NULL,
  stability_metric = "stability_score",
  title = "Distribution of Stability Scores"
)
```

Arguments

exposomicset	A MultiAssayExperiment object containing sensitivity analysis results in metadata(exposomicset
stability_score_thresh	A numeric threshold for stability scores. Default is NULL, which uses the threshold stored in metadata(exposomicset)$sensitivity_analysis$score_thresh.
stability_metric	A character string specifying which stability metric to plot (e.g., "stability_score", "logp_weighted_score"). Default is "stability_score".
title	A character string specifying the title of the ridge plot. Default is "Distribution of Stability Scores".

Details

This function:

- Extracts feature stability scores from metadata(exposomicset)$sensitivity_analysis$feature_stability.
- Displays a **ridge plot** of stability score distributions per assay.
- Displays a **bar chart** of the number of features per assay.
- Prints the number of features with stability scores above the threshold.

Value

A patchwork object combining a ridge plot and a bar chart.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)


# Run differential abundance
mae <- run_differential_abundance(
    exposomicset = mae,
    formula = ~ smoker + sex,
    abundance_col = "counts",
    method = "limma_voom",
    action = "add"
)

# Run the sensitivity analysis
mae <- run_sensitivity_analysis(
    exposomicset = mae,
    base_formula = ~ smoker + sex,
    methods = c("limma_voom"),
    scaling_methods = c("none"),
    covariates_to_remove = "sex",
    pval_col = "P.Value",
    logfc_col = "logFC",
    pval_threshold = 0.05,
    logFC_threshold = 0,
    bootstrap_n = 3,
```

```
    action = "add"
)

# create the sensitivity summary plot
sens_sum_p <- mae |>
    plot_sensitivity_summary()
```

`plot_top_factor_features` *Plot Top Features by Factor from Integration Results*

Description

Visualizes the top loading features for each factor from multi-omics integration results (e.g., MOFA, MCIA, DIABLO, RGCCA).

Usage

```
plot_top_factor_features(
  exposomicset,
  feature_col = "feature",
  factors = NULL,
  top_n = 5,
  facet_cols = NULL,
  exp_name_cols = NULL,
  alpha = 0.5
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing integration results in the `metadata` slot (must include `integration_results`).
`feature_col`	A character string indicating the column name to use for y-axis feature labels (e.g., `"feature"`, `"gene_symbol"`). This should match a column in the output of `pivot_feature()`. Default is `"feature"`.
`factors`	Character vector of factors to include (e.g., "Factor1", "Factor2"). If `NULL`, all factors are plotted.
`top_n`	Integer specifying the number of top features to show per factor. Default is `5`.
`facet_cols`	Optional color palette for facet strip backgrounds (one per `exp_name`), used to distinguish factors.
`exp_name_cols`	Optional color palette for experiment labels in the plot (`exp_name`), passed to `scale_color_manual()`.
`alpha`	Numeric value between 0 and 1 controlling the transparency of facet strip background fill. Default is `0.5`.

Details

This function supports the following integration methods:

- "MOFA": Uses feature weights from MOFA2 (get_weights()).
- "MCIA": Uses block loadings from MCIA (@block_loadings).
- "DIABLO": Extracts block-specific loadings from loadings.
- "RGCCA": Extracts block-specific loadings from a.

For each factor, it:

- Selects the top top_n features by **absolute loading**.
- Merges with feature metadata using pivot_feature().
- Creates a point-range plot showing the loading magnitude.
- Facets each factor with a customizable strip background.

The feature_col argument allows you to control which feature-level metadata column (e.g., gene symbols, metabolite names) is used for labeling the y-axis.

If palettes are not provided, defaults are chosen using ggpubr::get_palette().

Value

A ggplot2 object with one facet per factor, showing the top features and their loadings by experiment.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

mae <- run_multiomics_integration(
    mae,
    method = "DIABLO",
    outcome = "smoker",
    n_factors = 3
)

# plot top features using default `feature` column
top_feature_p <- mae |>
    plot_top_factor_features()
```

plot_volcano *Volcano Plot of Differential Abundance*

Description

Generates a **volcano plot** to visualize differential abundance results across one or more omics layers.

Usage

```
plot_volcano(
  exposomicset,
  pval_col = "adj.P.Val",
  pval_thresh = 0.05,
  logFC_col = "logFC",
  logFC_thresh = log2(1.5),
  plot_n_sig = TRUE,
  top_n_label = NULL,
  features_to_label = NULL,
  feature_col = "feature",
  xlab = expression(Log[2] * "FC"),
  ylab = expression(-Log[10] * "P"),
  title = "Volcano Plot of Differential Abundance",
  nrow = 2
)
```

Arguments

exposomicset	A MultiAssayExperiment object containing differential abundance results in metadata(exposomicset)$differential_abundance.
pval_col	A character string specifying the column containing p-values. Default is "adj.P.Val".
pval_thresh	A numeric threshold for significance. Features with p-values below this are considered significant. Default is 0.05.
logFC_col	A character string specifying the column for log fold changes. Default is "logFC".
logFC_thresh	A numeric threshold for absolute log fold change significance. Default is log2(1.5).
plot_n_sig	Logical; if TRUE, appends the number of significant features to facet titles. Default is TRUE.
top_n_label	Optional integer. If provided, the top n most significant features per assay will be labeled on the plot.
features_to_label	Optional character vector. Specific features to label regardless of significance.
feature_col	A character string naming the feature ID column to use for labeling. Default is "feature".
xlab	Label for the x-axis. Default is expression(Log[2]*"FC").
ylab	Label for the y-axis. Default is expression(-Log[10]*"P").
title	Plot title. Default is "Volcano Plot of Differential Abundance".
nrow	Number of rows in the facet_wrap() layout. Default is 2.

Details

The function:

- Extracts differential abundance results from `metadata(exposomicset)$differential_abundance`.
- Assigns each feature a direction of change: **Upregulated**, **Downregulated**, or **Not-Significant**.
- Uses `logFC_thresh` and `pval_thresh` to define thresholds.
- Adds dashed lines to indicate cutoffs for fold change and significance.
- Uses `facet_wrap()` to display each assay (`exp_name`) separately.
- Optionally labels the most significant features or user-defined ones.

Value

A `ggplot2` object representing the volcano plot.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# perform differential abundance analysis
mae <- run_differential_abundance(
    exposomicset = mae,
    formula = ~ smoker + sex,
    abundance_col = "counts",
    method = "limma_voom",
    action = "add"
)

# create the volcano plot
volcano_p <- mae |>
    plot_volcano()
```

`run_association` *Run Association Analysis*

Description

Perform GLM-based association testing between a specified outcome and features from exposures, omics, or latent factors. Automatically adjusts for covariates and supports both Gaussian and binomial models.

Usage

```
run_association(
  exposomicset,
  outcome,
  source = c("omics", "exposures", "factors"),
  covariates = NULL,
  feature_set = NULL,
  log_trans = TRUE,
  top_n = NULL,
  family = "gaussian",
  correction_method = "fdr",
  action = "add",
  feature_col = NULL,
  mirna_assays = NULL
)
```

Arguments

exposomicset	A MultiAssayExperiment object containing data and metadata.
outcome	The outcome variable name (must be in colData).
source	Source of features to test. One of "omics", "exposures", "factors".
covariates	Optional vector of covariate names to include in the model.
feature_set	Optional character vector of exposure or GO terms to test.
log_trans	Optional boolean value dictating whether or not to log2 transform omics features.
top_n	Optional integer: if using omics source, select top n most variable features.
family	GLM family; "gaussian" or "binomial".
correction_method	Method for p-value adjustment (default: "fdr").
action	If "add" (default), saves results to metadata; else returns results as list.
feature_col	The column in rowData for matching gene symbols or IDs.
mirna_assays	Optional character vector of assays to exclude when extracting GO terms.

Value

If action = "add", returns updated MultiAssayExperiment. Otherwise, returns a list of:

- results_df: tidy summary of associations
- covariates: the covariates used
- model_data: model matrix used in the GLMs

Examples

```
#' # create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# run association tests
```

```
mae <- mae |>
    run_association(
        source = "exposures",
        feature_set = c("exposure_pm25", "exposure_no2"),
        outcome = "smoker",
        covariates = c("age"),
        family = "binomial"
    )
```

`run_cluster_samples` *Cluster Samples Based on Exposure Data*

Description

Performs hierarchical clustering of samples using exposure data from `colData(exposomicset)`.

Usage

```
run_cluster_samples(
  exposomicset,
  exposure_cols = NULL,
  dist_method = NULL,
  user_k = NULL,
  cluster_method = "ward.D",
  clustering_approach = "diana",
  action = "add"
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing omics and exposure data.
`exposure_cols`	A character vector of column names in `colData(exposomicset)` to use for clustering.
`dist_method`	A character string specifying the distance metric (`"euclidean"`, `"gower"`, etc.). If `NULL`, it is automatically determined.
`user_k`	An integer specifying the number of clusters. If `NULL`, an optimal `k` is determined.
`cluster_method`	A character string specifying the hierarchical clustering method. Default is `"ward.D"`.
`clustering_approach`	A character string specifying the method for determining `k` (`"diana"`, `"gap"`, `"elbow"`, `"dynamic"`, or `"density"`). Default is `"diana"`.
`action`	A character string specifying `"add"` (store results in metadata) or `"get"` (return clustering results). Default is `"add"`.

Details

This function:

- Extracts **numeric exposure data** from `colData(exposomicset)`.

- Computes a **distance matrix** ("gower" for mixed data, "euclidean" for numeric).
- Determines the **optimal number of clusters** (k) using the specified method.
- Performs **hierarchical clustering** (hclust) and assigns samples to clusters.
- Generates a **heatmap** of scaled exposure values.
- Stores results in metadata(exposomicset)$sample_clustering when action="add".

Value

If action="add", returns the updated exposomicset. If action="get", returns a list with:

sample_cluster	A hierarchical clustering object (hclust).
sample_groups	A named vector of sample cluster assignments.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# determine sample clusters
mae <- run_cluster_samples(
    exposomicset = mae,
    exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi"),
    clustering_approach = "diana"
)
```

---

run_correlation *Run Correlation Analysis*

Description

Computes correlations between exposures and feature types including DEGs, omics, latent factors, top factor features, or principal components (PCs). Optionally computes feature-feature correlations to support network analysis.

Usage

```
run_correlation(
  exposomicset,
  feature_type = c("degs", "omics", "factors", "factor_features", "exposures", "pcs"),
  exposure_cols = NULL,
  variable_map = NULL,
  n_pcs = NULL,
  feature_cors = FALSE,
  robust = FALSE,
  score_col = "stability_score",
  score_thresh = NULL,
  correlation_method = "spearman",
```

```
  correlation_cutoff = 0.3,
  cor_pval_column = "p.value",
  pval_cutoff = 0.05,
  deg_pval_col = "adj.P.Val",
  deg_logfc_col = "logFC",
  deg_pval_thresh = 0.05,
  deg_logfc_thresh = log2(1.5),
  batch_size = 1500,
  action = c("add", "get")
)
```

Arguments

exposomicset	A MultiAssayExperiment object.
feature_type	Type of features to correlate. One of "degs", "omics", "factors", "factor_features", "exposures", or "pcs".
exposure_cols	Optional vector of exposure column names (from colData) to use.
variable_map	Optional mapping of features to include by assay for omics mode.
n_pcs	Number of PCs to use when feature_type = "pcs".
feature_cors	Logical; if TRUE, compute correlations between features rather than with exposures.
robust	Logical; restrict DEGs to those passing sensitivity threshold.
score_col	Column name in sensitivity analysis with feature stability score.
score_thresh	Threshold for filtering robust features.
correlation_method	One of "pearson", "spearman", or "kendall".
correlation_cutoff	Minimum absolute correlation to retain.
cor_pval_column	Column in output to filter by p-value (default: "p.value").
pval_cutoff	Maximum p-value or FDR threshold to retain a correlation.
deg_pval_col	Column with DEG adjusted p-values.
deg_logfc_col	Column with DEG log fold-changes.
deg_pval_thresh	P-value cutoff for DEGs.
deg_logfc_thresh	Log fold-change cutoff for DEGs.
batch_size	Number of features to process per batch (default: 1500).
action	Whether to "add" results to metadata or "get" as a data frame.

Value

If action = "add", returns updated MultiAssayExperiment with results added to metadata. If action = "get", returns a tidy data.frame of correlations.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# run correlation analysis
mae <- mae |>
    run_correlation(
        feature_type = "exposures",
        exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi")
    )
```

---

run_create_network *Create Correlation Network from Feature Data*

Description

Constructs an undirected feature-feature or feature-exposure correlation network from correlation results stored in a `MultiAssayExperiment` object. The function supports multiple correlation formats depending on `feature_type`, and stores or returns an `igraph` object with associated node and edge metadata.

Usage

```
run_create_network(
  exposomicset,
  feature_type = c("degs", "omics", "factors", "factor_features", "exposures",
    "degs_feature_cor", "omics_feature_cor", "factor_features_feature_cor"),
  action = c("add", "get")
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing correlation results in metadata.
`feature_type`	Type of correlation result to convert to a network. One of: `"degs"`, `"omics"`, `"factors"`, `"factor_features"`, `"exposures"`, `"degs_feature_cor"`, `"omics_feature_cor"`, or `"factor_features_feature_cor"`.
`action`	Whether to `"add"` the network to the object or `"get"` it as a list.

Details

The function detects the appropriate edge and node structure based on column names in the correlation results. Edge weights are based on correlation coefficients and include FDR values.

Value

If `action = "add"`, returns the updated `MultiAssayExperiment` with a new `network` entry in metadata. If `action = "get"`, returns a list with `graph` (an `igraph` object) and `summary` (a tibble).

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# perform correlation analyses
# correlate with exposures
mae <- mae |>
    run_correlation(
        feature_type = "omics",
        variable_map = mae |>
            pivot_feature() |>
            dplyr::select(
                variable = .feature,
                exp_name = .exp_name
            ),
        exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi")
    ) |>
    # correlate omics features with themselves
    run_correlation(
        feature_type = "omics",
        variable_map = mae |>
            pivot_feature() |>
            dplyr::select(
                variable = .feature,
                exp_name = .exp_name
            ),
        feature_cors = TRUE,
        exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi")
    )

# create the networks
mae <- mae |>
    run_create_network(
        feature_type = "omics_feature_cor",
        action = "add"
    ) |>
    run_create_network(
        feature_type = "omics",
        action = "add"
    )
```

`run_differential_abundance`

Run Differential Abundance Analysis

Description

Performs differential abundance testing across all assays in a `MultiAssayExperiment` object using a specified statistical method. The function updates each assay with its corresponding `colData`, fits the model using the provided formula, and combines the results into a unified table.

Usage

```
run_differential_abundance(
  exposomicset,
  formula,
  abundance_col = "counts",
  method = "limma_trend",
  contrasts = NULL,
  scaling_method = "none",
  action = "add"
)
```

Arguments

exposomicset	A MultiAssayExperiment containing assays to test.
formula	A model formula for the differential analysis (e.g., ~ group + batch).
abundance_col	Character. The name of the assay matrix to use for abundance values. Default is "counts".
method	Character. Differential analysis method to use. Currently supports "limma_trend" (default).
contrasts	A named list of contrasts for pairwise comparisons. Default is NULL (uses default group comparisons).
scaling_method	Character. Scaling method to apply before modeling. Options include "none" (default), "zscore", etc.
action	Character. Whether to "add" results to exposomicset metadata or "get" the results as a data frame. Default is "add".

Value

Either the updated MultiAssayExperiment (if action = "add") or a tibble with differential abundance results (if action = "get").

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# perform differential abundance analysis
mae <- run_differential_abundance(
    exposomicset = mae,
    formula = ~ smoker + sex,
    abundance_col = "counts",
    method = "limma_trend",
    action = "add"
)
```

run_enrichment *Perform enrichment analysis on selected features from a exposomicset object*

Description

This function performs enrichment analysis using selected features derived from differential expression, correlation analysis, or multi-omics factor features across experiments in an exposomicset. It supports multiple enrichment databases (e.g., GO, KEGG, Reactome), applies FDR correction, and optionally clusters GO terms by Jaccard overlap.

Usage

```
run_enrichment(
  exposomicset,
  feature_type = c("degs", "degs_robust", "omics", "factor_features", "degs_cor",
    "omics_cor", "factor_features_cor"),
  score_col = "stability_score",
  score_threshold = NULL,
  variable_map = NULL,
  factor_type = c("common_top_factor_features", "top_factor_features"),
  feature_col = "feature",
  deg_pval_col = "adj.P.Val",
  deg_pval_threshold = 0.05,
  deg_logfc_col = "logFC",
  deg_logfc_threshold = log2(1.5),
  db = c("GO", "KEGG", "Reactome"),
  species = NULL,
  fenr_col = "gene_symbol",
  padj_method = "fdr",
  pval_thresh = 0.1,
  min_set = 3,
  max_set = 800,
  clustering_approach = "diana",
  action = "add"
)
```

Arguments

exposomicset	An exposomicset (a MultiAssayExperiment object with metadata) containing omics and metadata.
feature_type	Character string indicating the feature source. One of "degs", "degs_robust", "omics", "factor_features", "degs_cor", "omics_cor", or "factor_features_cor".
score_col	Column name used for stability score filtering (only for degs_robust).
score_threshold	Optional numeric threshold for filtering stability scores. If NULL, the default threshold stored in the metadata will be used.
variable_map	A data frame with exp_name and variable columns, used when feature_type = "omics".

factor_type	Character string for selecting factor features: "common_top_factor_features" or "top_factor_features".
feature_col	The name of the feature column used to extract gene identifiers.
deg_pval_col	Column name for adjusted p-values from DEG analysis.
deg_pval_threshold	Threshold to select significant DEGs (default: 0.05).
deg_logfc_col	Column name for log-fold changes from DEG analysis.
deg_logfc_threshold	Threshold to select DEGs by absolute logFC (default: log2(1.5)).
db	Enrichment database to use. One of "GO", "KEGG", "Reactome".
species	Species name (required for GO enrichment, e.g., "Homo sapiens"). Ignored for other databases.
fenr_col	Column name for gene IDs used by fenr (e.g., "gene_symbol").
padj_method	Method for p-value adjustment (default: "fdr").
pval_thresh	Adjusted p-value threshold for filtering enriched terms (default: 0.1).
min_set	Minimum number of selected genes overlapping an enriched term (default: 3).
max_set	Maximum number of selected genes overlapping an enriched term (default: 800).
clustering_approach	Clustering method for GO term grouping. Defaults to "diana".
action	Either "add" to store results in the object's metadata or "get" to return results as a data frame.

Details

The function identifies selected features based on the chosen feature_type, determines the gene universe for each experiment, and performs enrichment analysis using the fenr package. Results are adjusted for multiple testing and optionally clustered by gene set overlap (for GO terms).

If feature_type includes correlation-based results (ending in _cor), enrichment is performed for each exposure category separately.

Value

If action = "add", returns the modified exposomicset with enrichment results added to metadata. If action = "get", returns a data.frame of enrichment results with GO term clusters (if applicable).

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 30,
    return_mae = TRUE
)

# perform differential abundance analysis
mae <- run_differential_abundance(
    exposomicset = mae,
    formula = ~ smoker + sex,
    abundance_col = "counts",
```

```
    method = "limma_voom",
    action = "add"
)

# perform enrichment analysis
mae <- run_enrichment(
    exposomicset = mae,
    feature_type = "degs",
    feature_col = "symbol",
    species = "goa_human",
    deg_logfc_threshold = log2(1),
    deg_pval_col = "P.Value",
    deg_pval_threshold = 0.2,
    action = "add"
)
```

`run_exposome_score` *Compute Composite Exposome Scores*

Description

Calculates a summary exposome score per sample using one of several methods including mean, sum, median, PCA (first principal component), IRT (Item Response Theory), quantile binning, or row-wise variance. The resulting score is added to the `colData` of the `MultiAssayExperiment` object.

Usage

```
run_exposome_score(
  exposomicset,
  score_type,
  exposure_cols = NULL,
  scale = TRUE,
  score_column_name = NULL
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing exposure data in its `colData`.
`score_type`	Character. The method used to compute the score. Options are: `"mean"`, `"sum"`, `"median"`, `"pca"`, `"irt"`, `"quantile"`, `"var"`.
`exposure_cols`	Optional character vector. Specific exposure column names to include. If `NULL`, all numeric columns are used.
`scale`	Logical. Whether to scale the exposures before computing the score. Default is `TRUE`.
`score_column_name`	Optional name for the resulting score column. If `NULL`, an automatic name is used (e.g., `"exposome_score_pca"`).

Details

- "pca" uses the first principal component from prcomp().
- "irt" uses the mirt package to fit a graded response model to discretized exposures.
- "quantile" assigns decile bins (1-10) to each variable and sums them row-wise.
- "var" computes the row-wise variance across exposures.

Value

A MultiAssayExperiment object with the exposome score added to colData().

Examples

```
# create the example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# create the air pollution score
mae <- run_exposome_score(
    mae,
    score_type = "pca",
    exposure_cols = c("exposure_pm25", "exposure_no2"),
    scale = TRUE,
    score_column_name = "air_pollution_score"
)
```

---

run_exposure_impact *Calculate Exposure Impact from Feature-Exposure Correlation Networks*

Description

Generalized centrality-based exposure impact analysis using DEG, omics, or factor features.

Usage

```
run_exposure_impact(
  exposomicset,
  feature_type = c("degs", "omics", "factor_features"),
  pval_col = "adj.P.Val",
  pval_thresh = 0.1,
  action = c("add", "get")
)
```

Arguments

exposomicset	A MultiAssayExperiment object with correlation and network metadata.
feature_type	One of "degs", "omics", or "factor_features".
pval_col	Column in differential abundance results to filter DEGs. Default = "adj.P.Val".
pval_thresh	DEG p-value threshold. Ignored unless feature_type == "degs".
action	Either "add" (store in metadata) or "get" (return list).

Value

Either an updated MultiAssayExperiment (if action = "add") or a list.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# perform correlation analyses
# correlate with exposures
mae <- mae |>
    run_correlation(
        feature_type = "omics",
        variable_map = mae |>
            pivot_feature() |>
            dplyr::select(
                variable = .feature,
                exp_name = .exp_name
            ),
        exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi")
    ) |>
    run_correlation(
        feature_type = "omics",
        variable_map = mae |>
            pivot_feature() |>
            dplyr::select(
                variable = .feature,
                exp_name = .exp_name
            ),
        feature_cors = TRUE,
        exposure_cols = c("exposure_pm25", "exposure_no2", "age", "bmi")
    )

# create the networks
mae <- mae |>
    run_create_network(
        feature_type = "omics_feature_cor",
        action = "add"
    ) |>
    run_create_network(
        feature_type = "omics",
        action = "add"
    )

# perform impact analysis
mae <- mae |>
    run_exposure_impact(
        feature_type = "omics"
    )
```

`run_exposure_omics_association` *Run Exposure-Omics Association*

Description

Test associations between each exposure and each omics feature using limma's linear modeling framework.

Usage

```
run_exposure_omics_association(
  exposomicset,
  exposures = NULL,
  exp_name = NULL,
  covariates = NULL,
  scaling_method = "none",
  correction_method = "fdr",
  top_pct = NULL,
  filter_by = c("variance", "mean"),
  action = "add"
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing exposome and omics data.
`exposures`	Character vector of exposure variable names to test. If `NULL`, uses all variables from the codebook.
`exp_name`	Name(s) of the omics assay(s) to test against. If `NULL`, uses all assays.
`covariates`	Optional character vector of covariate names to include in the model.
`scaling_method`	Character. Scaling method to apply before modeling. Options include `"none"` (default).
`correction_method`	Method for p-value adjustment. Default is `"fdr"`.
`top_pct`	Top X% of features to retain using either mean or variance which is specified by `filter_by`. If `NULL`, no features will be filtered.
`filter_by`	Determination of how to filter omics features either by mean or variance.
`action`	If `"add"` (default), saves results to metadata, if `"get"`, returns results as a data frame.

Details

This function uses limma to test associations between multiple exposures and omics features. For each exposure, a linear model is fit with the exposure as the predictor and each omics feature as the outcome, adjusting for covariates.

`omics_feature ~ exposure + covariate1 + covariate2 + ...`

Value

If `action = "add"`, returns updated `MultiAssayExperiment`. Otherwise, returns a tibble with association results.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# run exposure-omics association
mae <- mae |>
    run_exposure_omics_association(
        exposures = c("exposure_pm25", "exposure_no2"),
        covariates = c("age", "sex")
    )
```

---

run_factor_overlap *Identify and Annotate Shared Top Features Across Integration Factors*

Description

Identifies top features shared across factors based on integration method. For MOFA/MCIA, takes intersection across factors. For DIABLO/RGCCA, takes features recurring in more than 2 block-specific components.

Usage

```
run_factor_overlap(
  exposomicset,
  robust = TRUE,
  stability_score = NULL,
  score_col = "stability_score",
  pval_thresh = 0.05,
  logfc_thresh = log2(1.5),
  pval_col = "padj",
  logfc_col = "logFC",
  action = "add"
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` with integration results and top factor features.
`robust`	Logical; if `TRUE`, uses sensitivity score. Otherwise, uses DEG thresholds.
`stability_score`	Optional numeric threshold (overrides default from metadata).
`score_col`	Column name for sensitivity score. Default is `"stability_score"`.
`pval_thresh`	DEG p-value threshold (if `robust = FALSE`). Default is `0.05`.

logfc_thresh	DEG logFC threshold (if robust = FALSE). Default is log2(1.5).
pval_col	Column name for p-value. Default is "padj".
logfc_col	Column name for logFC. Default is "logFC".
action	"add" to return modified object, "get" to return data.frame.

Value

Modified MultiAssayExperiment or data.frame of shared top features.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# perform multiomics integration
mae <- run_multiomics_integration(
    mae,
    method = "DIABLO",
    outcome = "smoker",
    n_factors = 3
)


# identify the features that contribute most to the factors
mae <- extract_top_factor_features(
    mae,
    factors = c("V1", "V2", "V3"),
    method = "percentile",
    percentile = 0.5,
    action = "add"
)

# perform differential abundance analysis
mae <- run_differential_abundance(
    exposomicset = mae,
    formula = ~ smoker + sex,
    abundance_col = "counts",
    method = "limma_voom",
    action = "add"
)

# determine the overlap in features
mae <- mae |>
    run_factor_overlap(
        robust = FALSE,
        pval_col = "adj.P.Val"
    )
```

`run_impute_missing` *Impute Missing Exposure and Omics Data in a MultiAssayExperiment*

Description

Performs missing data imputation on both exposure variables (from `colData`) and omics datasets (from `experiments`) within a `MultiAssayExperiment` object.

Usage

```
run_impute_missing(
  exposomicset,
  exposure_impute_method = "median",
  exposure_cols = NULL,
  omics_impute_method = NULL,
  omics_to_impute = NULL
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing exposures and omics data.
`exposure_impute_method`	Character. Imputation method to use for exposure variables. Defaults to `"median"`.
`exposure_cols`	Character vector. Names of columns in `colData` to impute. If `NULL`, all numeric columns are used.
`omics_impute_method`	Character. Imputation method to use for omics data. Defaults to `"knn"`.
`omics_to_impute`	Character vector. Names of omics datasets to impute. If `NULL`, all omics datasets are included.

Details

For exposures, numeric columns in `colData` are imputed using the selected method. For omics data, assays are selected and imputed individually.

Supported imputation methods include:

- `"median"`: Median imputation using `naniar::impute_median_all`
- `"mean"`: Mean imputation using `naniar::impute_mean_all`
- `"knn"`: k-nearest neighbor imputation using `impute::impute.knn`
- `"mice"`: Multiple imputation using chained equations (`mice::mice`)
- `"missforest"`: Random forest-based imputation using `missForest::missForest`
- `"lod_sqrt2"`: Substitution of missing values with LOD/sqrt(2), where LOD is the smallest non-zero value per variable

Value

A `MultiAssayExperiment` object with imputed exposure and/or omics data.

Examples

```
# Create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# Introduce some missingness
MultiAssayExperiment::colData(mae)$exposure_pm25[sample(1:20, 5)] <- NA

# Filter features and exposures with high missingness
mae <- run_impute_missing(
    exposomicset = mae,
    exposure_impute_method = "median"
)
```

`run_multiomics_integration` *Run Multi-Omics Integration*

Description

Performs multi-omics integration using one of several available methods: MOFA, MCIA, RGCCA, or DIABLO. This function takes a `MultiAssayExperiment` object with two or more assays and computes shared latent factors across omics layers.

Usage

```
run_multiomics_integration(
  exposomicset,
  method = "MCIA",
  n_factors = 10,
  scale = TRUE,
  outcome = NULL,
  max.iter = 500,
  near.zero.var = TRUE,
  action = "add"
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object with at least two assays.
`method`	Character. Integration method to use. Options are `"MOFA"`, `"MCIA"`, `"RGCCA"`, or `"DIABLO"`.
`n_factors`	Integer. Number of latent factors/components to compute. Default is 10.
`scale`	Logical. Whether to scale each assay before integration. Default is `TRUE`.
`outcome`	Character. Required if `method = "DIABLO"`. Name of outcome variable in `colData` used for supervised integration.
`max.iter`	numeric. Option to increase the number of iterations for `mixOmics::block.splsda` the default is 500.

near.zero.var Logical. Option to remove variables with near zero variance for mixOmics::block.splsda, default is TRUE .

action Character. Whether to "add" results to the metadata or "get" them as a list. Default is "add".

Details

- "MOFA" runs Multi-Omics Factor Analysis using the MOFA2 package and returns a trained model.
- "MCIA" runs multi-co-inertia analysis using the nipalsMCIA package.
- "RGCCA" runs Regularized Generalized Canonical Correlation Analysis using the RGCCA package.
- "DIABLO" performs supervised integration using the mixOmics package and a specified outcome.

Value

If action = "add", returns a MultiAssayExperiment with integration results stored in metadata(exposomicset)$mult
If action = "get", returns a list with integration method and result.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# perform multiomics integration
mae <- run_multiomics_integration(
    mae,
    method = "DIABLO",
    outcome = "smoker",
    n_factors = 3
)
```

---

run_normality_check *Assess Normality of Exposure Variables*

Description

Performs Shapiro-Wilk tests to check the normality of numeric exposure variables in colData of a MultiAssayExperiment object.

Usage

```
run_normality_check(exposomicset, action = "add")
```

Arguments

exposomicset	A MultiAssayExperiment object containing exposure data in colData.
action	A character string specifying whether to store ("add") or return ("get") the results. Default is "add".

Details

This function:

- Extracts **numeric, non-constant** exposure variables from colData.
- Runs **Shapiro-Wilk tests** to assess normality.
- Summarizes the number of normally and non-normally distributed exposures.
- Generates a bar plot visualizing the normality results.
- **Output Handling**:
  - "add": Stores results in metadata(exposomicset)$normality.
  - "get": Returns a list containing the normality test results and plot.

Value

A MultiAssayExperiment object with normality results added to metadata (if action = "add") or a list with:

norm_df	A data frame of Shapiro-Wilk test results for each exposure variable.
norm_plot	A ggplot object showing the distribution of normal and non-normal exposures.

Examples

```
# Create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# Test for normality
mae <- mae |>
    run_normality_check() |>
    transform_exposure(exposure_cols = c("age", "bmi", "exposure_pm25"))
```

---

run_pca *Perform Principal Component Analysis (PCA)*

Description

Runs PCA on the feature and sample spaces of a MultiAssayExperiment object, identifying outliers based on Mahalanobis distance.

Usage

```
run_pca(
  exposomicset,
  log_trans_exp = FALSE,
  log_trans_omics = TRUE,
  action = "add"
)
```

Arguments

exposomicset	A MultiAssayExperiment object containing omics and exposure data.
log_trans_exp	A boolean value specifying whether to log2 transform the exposure data
log_trans_omics	a boolean value specifying whether to log2 transform the omics data
action	A character string specifying whether to store ("add") or return ("get") the results. Default is "add".

Details

This function:

- Identifies **common samples** across all assays and exposure data.
- Performs **PCA on features** (transformed and standardized).
- Performs **PCA on samples** and computes Mahalanobis distance to detect outliers.

Value

If action = "add", the function returns the input MultiAssayExperiment with:

- **PC scores** added as columns in colData(exposomicset), and
- **PCA objects** stored under metadata(exposomicset)$quality_control$pca.

If action = "get", the function returns a list containing:

pca_df	A tibble of the transformed input data.
pca_feature	A prcomp object containing PCA results for features.
pca_sample	A prcomp object containing PCA results for samples.
outliers	A character vector of detected sample outliers.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 10,
    return_mae = TRUE
)

# run pca
mae <- mae |>
    run_pca()
```

`run_pipeline_summary` *Summarize and Visualize Analysis Pipeline Steps*

Description

This function prints and visualizes the analysis steps stored in the metadata of a `MultiAssayExperiment` object. The steps are optionally printed to the console as a numbered list and can be rendered as a left-to-right Mermaid flowchart. The flowchart connects steps with arrows and includes step notes if requested.

Usage

```
run_pipeline_summary(
  exposomicset,
  show_index = TRUE,
  console_print = TRUE,
  diagram_print = FALSE,
  include_notes = TRUE
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object that contains a "summary" entry in its metadata, which includes a list named `steps`.
`show_index`	Logical, default `TRUE`. If `TRUE`, prefixes each step with its index.
`console_print`	Logical, default `TRUE`. If `TRUE`, prints the step list to the console.
`diagram_print`	Logical, default `FALSE`. If `TRUE`, renders a Mermaid diagram of the steps.
`include_notes`	Logical, default `TRUE`. If `TRUE`, appends any "notes" associated with each step to the label.

Details

The Mermaid flowchart is rendered left-to-right and connects each step in sequence. Each node is labeled using the step name and, optionally, any attached notes.

Value

No return value. This function is called for its side effects: console output and/or diagram rendering.

Examples

```
# Create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# Test for normality
mae <- mae |>
    run_normality_check() |>
    transform_exposure(exposure_cols = c("age", "bmi", "exposure_pm25"))
```

```
# Run the pipeline summary
run_pipeline_summary(mae)
```

`run_sensitivity_analysis` *Run Sensitivity Analysis for Differential Abundance*

Description

Performs sensitivity analysis by systematically varying statistical methods, scaling strategies, and model formulas (with optional bootstrap sampling) to assess the stability of differentially abundant features.

Usage

```
run_sensitivity_analysis(
  exposomicset,
  base_formula,
  abundance_col = "counts",
  methods = c("limma_trend", "limma_voom", "DESeq2", "edgeR_quasi_likelihood"),
  scaling_methods = c("none", "TMM", "quantile"),
  contrasts = NULL,
  covariates_to_remove = NULL,
  pval_col = "adj.P.Val",
  logfc_col = "logFC",
  pval_threshold = 0.05,
  logFC_threshold = log2(1),
  score_thresh = NULL,
  score_quantile = 0.9,
  stability_metric = "stability_score",
  action = "add",
  bootstrap_n = 1
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` containing the assays to analyze.
`base_formula`	The base model formula used for differential analysis.
`abundance_col`	Character. Name of the column in the assays representing abundance. Default is `"counts"`.
`methods`	Character vector of differential expression methods. Options include `"limma_trend"` ,`"limma_voom"`, `"DESeq2"`, and `"edgeR_quasi_likelihood"`.
`scaling_methods`	Character vector of normalization methods to try. Options include `"none"`, `"TMM"`, and `"quantile"`.
`contrasts`	Optional list of contrasts to apply for differential testing.

`covariates_to_remove`
: Optional character vector of covariates to remove from the base formula to generate model variants.

`pval_col`
: Name of the column containing p-values or adjusted p-values used to define significance.

`logfc_col`
: Name of the column containing log fold changes.

`pval_threshold`
: Numeric threshold for significance. Default is 0.05.

`logFC_threshold`
: Numeric threshold for absolute log fold change. Default is `log2(1)` (i.e., 0).

`score_thresh`
: Optional threshold for the selected stability metric. If not provided, calculated using `score_quantile`.

`score_quantile`
: Quantile used to define the threshold if `score_thresh` is not provided. Default is 0.9.

`stability_metric`
: Character. Name of the column in `feature_stability` to use as the scoring metric. Default is `"stability_score"`.

`action`
: Whether to `"add"` results to `metadata()` or `"get"` them as a list. Default is `"add"`.

`bootstrap_n`
: Integer. Number of bootstrap iterations. If 0, no resampling is performed. Default is 1.

Value

If `action = "add"`, returns a `MultiAssayExperiment` with results stored in `metadata(exposomicset)$differential_`
If `action = "get"`, returns a list with three elements:

`sensitivity_df` Data frame of all differential results across model/method combinations.

`feature_stability` Data frame summarizing feature stability scores.

`score_thresh` The threshold used to define stable features.

Examples

```
# create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)


# Run differential abundance
mae <- run_differential_abundance(
    exposomicset = mae,
    formula = ~ smoker + sex,
    abundance_col = "counts",
    method = "limma_voom",
    action = "add"
)

# Run the sensitivity analysis
mae <- run_sensitivity_analysis(
    exposomicset = mae,
    base_formula = ~ smoker + sex,
```

```
    methods = c("limma_voom"),
    scaling_methods = c("none"),
    covariates_to_remove = "sex",
    pval_col = "P.Value",
    logfc_col = "logFC",
    pval_threshold = 0.05,
    logFC_threshold = 0,
    bootstrap_n = 3,
    action = "add"
)
```

`run_summarize_exposures`

Summarize Exposure Variables

Description

Computes summary statistics for numeric exposure variables and optionally stores the results in the `MultiAssayExperiment` metadata.

Usage

```
run_summarize_exposures(exposomicset, exposure_cols = NULL, action = "add")
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing exposure data in the sample metadata.
`exposure_cols`	A character vector of exposure variable names to summarize. If `NULL`, all numeric exposure variables are included.
`action`	A string specifying the action to take. Use `"add"` to attach the summary table to `metadata(exposomicset)` or `"get"` to return the summary table directly. Default is `"add"`.

Details

This function:

- Extracts sample-level exposure data using `pivot_sample()`.
- Filters to user-specified exposures (`exposure_cols`) if provided.
- Computes descriptive statistics for each numeric variable:
  - Number of values (`n_values`)
  - Number of NAs (`n_na`)
  - Minimum, maximum, and range
  - Sum, median, mean
  - Standard error of the mean
  - 95% confidence interval of the mean
  - Variance, standard deviation
  - Coefficient of variation (`sd / mean`)
- Merges the result with variable metadata stored in `metadata(exposomicset)$codebook`.

Value

A modified `MultiAssayExperiment` object (if `action = "add"`), or a data frame of summary statistics (if `action = "get"`).

Examples

```
# Create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# Summarize exposure data
exp_sum <- mae |>
    run_summarize_exposures(
        exposure_cols = c("age", "bmi", "exposure_pm25"),
        action = "get"
    )
```

---

tidyexposomics_example

Example exposome multi-omics dataset

Description

A downsampled version of the ISGlobal Exposome Data Challenge 2021 dataset (Maitre *et al.*, *Environment International*, 2022; DOI: 10.1016/j.envint.2022.107422). This example dataset is included with **tidyexposomics** for vignette evaluation. This subset represents samples from children with low socioeconomic status in the first cohort of the original dataset.

Usage

```
data("tidyexposomics_example")
```

Format

An object of class `list` of length 6.

Details

The data have been filtered and transformed to illustrate the `tidyexposomics` workflow. Only a small subset of variables, and the top 500 most variable features per omic layer are retained.

Contents

meta A data frame of selected exposure, demographic, and outcome variables for a subset of participants.

annotated_cb A data frame providing ontology-linked annotation for the exposures in `meta`.

exp_filt A numeric matrix of gene expression values (500 features, 48 samples) representing the top-variance transcripts.

exp_fdata Feature-level metadata for `exp_filt`, including cleaned gene symbols.

methyl_filt A numeric matrix of DNA methylation M-values (500 CpG sites, 48 samples).

methyl_fdata Feature-level metadata for `methyl_filt`.

Source

Derived from the ISGlobal Exposome Data Challenge 2021 (Maitre *et al.*, *Environment International*, 2022; DOI: 10.1016/j.envint.2022.107422),licensed under CC-BY 4.0. The original data are available on Figshare (Project 98813) and GitHub (`isglobal-exposomeHub/ExposomeDataChallenge2021`). This example dataset was processed and downsampled by the tidyexposomics authors and is not a replacement for the full dataset.

Examples

```
data("tidyexposomics_example")
```

---

transform_exposure *Transform Exposure Variables for Normality*

Description

Applies a transformation to selected numeric exposure variables in the `colData` of a `MultiAssayExperiment` to improve their normality (e.g., log, Box-Cox, sqrt). Transformation results and normality statistics are stored in metadata for tracking.

Usage

```
transform_exposure(
  exposomicset,
  exposure_cols = NULL,
  transform_method = "boxcox_best"
)
```

Arguments

`exposomicset`	A `MultiAssayExperiment` object containing exposure variables in `colData`.
`exposure_cols`	Optional character vector of exposure variable names to transform. If `NULL`, uses exposures found in `metadata(exposomicset)$quality_control$normality$norm_df$exposure`
`transform_method`	Character. Transformation method to apply. Options: • `"none"`: no transformation • `"log2"`: log base 2 transformation • `"x_1_3"`: cube-root transformation • `"sqrt"`: square-root transformation • `"boxcox_best"`: data-driven Box-Cox approximation with heuristic labeling

Details

For `transform_method = "boxcox_best"`, the function automatically shifts values to be strictly positive and chooses from a discrete set of transformations (e.g., `1/x`, `log(x)`, `x^2`) based on estimated Box-Cox lambda. Each variable may receive a different transformation.

Value

A `MultiAssayExperiment` object with transformed exposures in `colData`, and transformation details stored in:

- `metadata(exposomicset)$quality_control$transformation$norm_df`: Shapiro-Wilk test results
- `metadata(exposomicset)$quality_control$transformation$norm_summary`: Summary of normality
- `metadata(exposomicset)$codebook`: Updated with transformation info per variable
- `metadata(exposomicset)$summary$steps`: Updated with step record

See Also

`boxcox`, `shapiro.test`

Examples

```
# Create example data
mae <- make_example_data(
    n_samples = 20,
    return_mae = TRUE
)

# Test for normality
mae <- mae |>
    run_normality_check() |>
    transform_exposure(
        exposure_cols = c("age", "bmi", "exposure_pm25"),
        transform_method = "boxcox_best"
    )
```

Index