Package 'romic'

May 9, 2026

Type Package

Title R for High-Dimensional Molecular Data

Version 1.3.3

Description Represents high-dimensional data as tables of features, samples and measurements, and a design list for tracking the meaning of individual variables. Using this format, filtering, normalization, and other transformations of a dataset can be carried out in a flexible manner. 'romic' takes advantage of these transformations to create interactive 'shiny' apps for exploratory data analysis such as an interactive heatmap.

Depends R (>= 4.1)

Imports checkmate, cli, dplyr, ggplot2, glue, purrr, readr, reshape2, rlang, shiny (>= 1.5.0), stringr, tibble, tidyr (>= 1.0.0)

Suggests knitr, impute, lazyeval, rmarkdown, usethis, testthat (>= 3.0.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL https://www.shackett.org/external/romic-docs/, https://github.com/shackett/romic

BugReports https://github.com/shackett/romic/issues

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Sean Hackett [aut, cre] (ORCID: <https://orcid.org/0000-0002-9553-4341>), Calico Life Sciences LLC [cph, fnd]

Maintainer Sean Hackett <seanrhackett@gmail.com>

Repository CRAN

Date/Publication 2026-01-31 21:40:02 UTC

Contents

`add_pcs` *Add PCA Loadings*

Description

Add Principal Components Analysis Loadings to a tidy or triple omics dataset.

Usage

```
add_pcs(
  tomic,
  value_var = NULL,
  center_rows = TRUE,
  npcs = NULL,
  missing_val_method = "drop_samples",
  label_percent_varex = TRUE,
  verbose = TRUE
)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
value_var	An abundance value to use with hclust
center_rows	center rows before performing PCA
npcs	number of principal component loadings to add to samples (default is number of samples)
missing_val_method	Approach to remove missing values: **drop_features** Drop features with missing values **drop_samples** Drop samples which are missing all features, then drop features **impute** Impute missing values
label_percent_varex	If true then PCs will be labelled by the percent of variability they explain.
verbose	extra reporting messages

Value

A tomic object with principal components added to samples.

Examples

```
add_pcs(brauer_2008_triple, npcs = 5)
```

---

app_flow *Flow*

Description

Using shiny comb through datasets by iterating between plotting steps, and lassoing steps to select points of interest.

Usage

```
app_flow(tomic)
```

Arguments

tomic	Either a tidy_omic or triple_omic object

Value

A shiny app

Examples

```
if (interactive()) {
  # library(reactlog)
  # reactlog_enable()
  app_flow(brauer_2008_triple)
  # shiny::reactlogShow()
}
```

---

app_heatmap *Interactive Heatmap*

Description

Generate a shiny interactive heatmap that allows for on demand filtering, ordering and faceting by variables of interest.

Usage

```
app_heatmap(tomic)
```

Arguments

tomic	Either a tidy_omic or triple_omic object

Value

A shiny app

Examples

```
if (interactive()) {
  app_heatmap(brauer_2008_tidy)
}
```

---

app_pcs *PC Plot*

Description

Generate a Shiny interactive scatter plot which allows visualization of features, measurements, and samples (with principal components added).

Usage

```
app_pcs(tomic)
```

Arguments

tomic	Either a tidy_omic or triple_omic object

Value

A shiny app

Examples

```
if (interactive()) {
  app_pcs(brauer_2008_tidy)
}
```

---

brauer_2008 *Brauer 2008*

Description

An RNA expression (microarray) dataset looking at how yeast gene expression changes as nutrient sources and nutrient richness changes.

brauer_2008 formatted as a tidy_omic object

brauer_2008 formatted as a triple_omic object

Usage

```
brauer_2008

brauer_2008_tidy

brauer_2008_triple
```

Format

A tibble with 18,000 rows and 8 columns:

name Common gene name

BP Gene ontology biological process of the gene

MF Gene ontology molecular function of the gene

sample Sample name

nutrient Which nutrient limits growth (Glucose, Nitrogen, Phosphorous, Sulfur, Uracil, Leucine

DR Dilution rate of the culture - basically how fast the cells are growing

expression Expression level of the gene, log2 observation relative to a replicate of G0.3

An object of class tidy_omic (inherits from tomic, general) of length 2.

An object of class triple_omic (inherits from tomic, general) of length 4.

Details

This version of the dataset contains only 500 genes randomly selected from the ~6K genes in the complete dataset.

Source

https://pubmed.ncbi.nlm.nih.gov/17959824/

`calculate_sample_mahalanobis_distances`
Calculate Sample Mahalanobis Distances

Description

Determine each samples distance from the center of the data using Mahalanobis distance.

Usage

```
calculate_sample_mahalanobis_distances(
  tomic,
  value_var = NULL,
  max_pcs = 10,
  scale = FALSE
)
```

Arguments

`tomic`	Either a `tidy_omic` or `triple_omic` object
`value_var`	the measurement variable to use for calculating distances
`max_pcs`	the maximum number of principal components to used for representing the covariance matrix.
`scale`	if TRUE then the data will be scaled before calculating distances

Details

Since ‘romic‘ is built around using tall data where there are more features than samples calculating Mahalanobis distance off of the covariance matrix is not possible. Instead, we use SVD to create a low-dimensional representation of the covariance matrix and calculate distances from the center of the data in this space. This essentially involves weighting the principal components by their loadings.

Value

The samples tibble with a new column ‘pc_distance‘ which contains the Mahalanobis distances of individual samples from the PC elipsoid

Examples

```
calculate_sample_mahalanobis_distances(brauer_2008_tidy)
```

`center_tomic` *Center T* Omic*

Description

Center each measurement by subtracting the mean.

Usage

```
center_tomic(tomic, measurement_vars = "all")
```

Arguments

`tomic`	Either a `tidy_omic` or `triple_omic` object
`measurement_vars`	measurement variables to center

Value

A `tomic` object where one or more measurements have been centered on a feature-by-feature basis.

Examples

```
center_tomic(brauer_2008_tidy)
```

`check_design` *Check Design*

Description

Check that the design list embedded in ‘tomic‘ objects is properly formatted.

Usage

```
check_design(tomic_design)
```

Arguments

`tomic_design`	a list with named attributes describing feature, sample, and measurement variables.

Value

0, invisibly

Examples

```
check_design(brauer_2008_triple$design)
```

---

check_tidy_omic *Check Tidy Omic*

Description

Check a tidy omic dataset for consistency between the data and design and validate that the dataset follows the `tidy_omic`/`tomic` specification.

Usage

```
check_tidy_omic(tidy_omic, fast_check = TRUE)
```

Arguments

`tidy_omic`	an object of class tidy_omic produced by `create_tidy_omic`
`fast_check`	if TRUE then skip some checks which are slow and that are generally only needed when a `tomic` object is first created.

Value

0 invisibly

---

check_tomic *Check T*Omic*

Description

Check a tidy or triple 'omic object for common pathologies, such as a mismatch between data and schema and non-uniqueness of primary keys.

Usage

```
check_tomic(tomic, fast_check = TRUE)
```

Arguments

`tomic`	Either a `tidy_omic` or `triple_omic` object
`fast_check`	if TRUE then skip some checks which are slow and that are generally only needed when a `tomic` object is first created.

Value

0 invisibly

Examples

```
check_tomic(brauer_2008_triple)
```

`check_triple_omic` *Check Triple Omic*

Description

Check a triple omic dataset for consistency between the data and design and validate that the dataset follows the `triple_omic/tomic` specification.

Usage

```
check_triple_omic(triple_omic, fast_check = TRUE)
```

Arguments

`triple_omic`	an object of class triple_omic produced by `create_triple_omic`
`fast_check`	if TRUE then skip some checks which are slow and that are generally only needed when a `tomic` object is first created.

Value

0 invisibly

`convert_wide_to_tidy_omic`
Convert Wide to Tidy Omic

Description

Convert a wide dataset of species' abundances (gene product, metabolites, lipids, ...) into a triple_omic dataset (one observation per row)

Usage

```
convert_wide_to_tidy_omic(
  wide_df,
  feature_pk,
  feature_vars = NULL,
  sample_var = "sample",
  measurement_var = "abundance",
  omic_type_tag = "general",
  verbose = TRUE
)
```

Arguments

wide_df	a data.frame (or tibble) containing 1+ columns of feature attributes and many columns of samples
feature_pk	A unique identifier for features
feature_vars	a character vector of additional feature-level variables (or NULL if there are no additional variables)
sample_var	variable name to use for samples
measurement_var	variable name to use for measurements
omic_type_tag	an optional subtype of omic data: metabolomics, lipidomics, proteomics, genomics, general
verbose	extra reporting messages

Value

A `tidy_omic` object as produced by `create_tidy_omic`.

Examples

```
library(dplyr)

wide_measurements <- brauer_2008_triple[["measurements"]] %>%
  tidyr::pivot_wider(names_from = sample, values_from = expression)

wide_df <- brauer_2008_triple[["features"]] %>%
  left_join(wide_measurements, by = "name")

convert_wide_to_tidy_omic(wide_df,
  feature_pk = "name",
  feature_vars = c("BP", "MF", "systematic_name")
)
```

`create_tidy_omic` *Create Tidy Omic*

Description

A tidy omics object contains a formatted dataset and a summary of the experimental design.

Usage

```
create_tidy_omic(
  df,
  feature_pk,
  feature_vars = NULL,
  sample_pk,
  sample_vars = NULL,
  omic_type_tag = "general",
  verbose = TRUE
)
```

Arguments

`df`	a data.frame (or tibble) containing some combination of feature, sample and observation-level variables
`feature_pk`	A unique identifier for features
`feature_vars`	a character vector of additional feature-level variables (or NULL if there are no additional variables)
`sample_pk`	A unique identifier for samples
`sample_vars`	a character vector of additional sample-level variables (or NULL if there are no additional variables)
`omic_type_tag`	an optional subtype of omic data: metabolomics, lipidomics, proteomics, genomics, general
`verbose`	extra reporting messages

Value

An S3 `tidy_omic/tomic` object built on a `list`:

data A tibble with one row per measurement (i.e., features x samples)

design A list which organized the dataset's meta-data:

feature_pk variable specifying a unique feature

sample_pk variable specifying a unique sample

features tibble of feature attributes

samples tibble of sample attributes

measurements tibble of measurement attributes

Examples

```
library(dplyr)

measurement_df <- tidyr::expand_grid(
  feature_id = 1:10,
  sample_id = LETTERS[1:5]
) %>%
  dplyr::mutate(value = rnorm(n()))

feature_df <- tibble(
  feature_id = 1:10,
  feature_group = rep(c("a", "b"), each = 5)
)
sample_df <- tibble(
  sample_id = LETTERS[1:5],
  sample_group = c("a", "a", "b", "b", "b")
)

triple_omic <- create_triple_omic(
  measurement_df, feature_df, sample_df,
  "feature_id", "sample_id"
)
raw_tidy_omic <- triple_to_tidy(triple_omic)$data

create_tidy_omic(raw_tidy_omic,
  feature_pk = "feature_id",
  feature_vars = "feature_group", sample_pk = "sample_id",
  sample_vars = "sample_group"
)
```

---

create_triple_omic *Create Triple Omic*

Description

A triple omics class contains three data.frames, one for features, one for samples, and one for abundances. This is a good format when there is a large amount of meta data associated with features or samples.

Usage

```
create_triple_omic(
  measurement_df,
  feature_df = NULL,
  sample_df = NULL,
  feature_pk,
  sample_pk,
  omic_type_tag = "general"
)
```

Arguments

measurement_df	A data.frame (or tibble) of measurements - one row for each combination of feature and sample
feature_df	A data.frame (or tibble) of features - one row per feature
sample_df	A data.frame (or tibble) of samples - one row per sample
feature_pk	A unique identifier for features
sample_pk	A unique identifier for samples
omic_type_tag	an optional subtype of omic data: metabolomics, lipidomics, proteomics, genomics, general

Details

for now primary keys are unique (rather than allowing for a multi-index)

Value

An S3 `triple_omic/tomic` object built on a `list`:

features A tibble of feature meta-data (one row per feature)

samples A tibble of sample meta-data (one row per sample)

measurements A tibble with one row per measurement (i.e., features x samples)

design A list which organized the dataset's meta-data:

- **feature_pk** variable specifying a unique feature
- **sample_pk** variable specifying a unique sample
- **features** tibble of feature attributes
- **samples** tibble of sample attributes
- **measurements** tibble of measurement attributes

Examples

```
library(dplyr)

measurement_df <- tidyr::expand_grid(
  feature_id = 1:10,
  sample_id = LETTERS[1:5]
) %>%
  dplyr::mutate(value = rnorm(n()))

feature_df <- tibble(
  feature_id = 1:10,
  feature_group = rep(c("a", "b"), each = 5)
)
sample_df <- tibble(
  sample_id = LETTERS[1:5],
  sample_group = c("a", "a", "b", "b", "b")
)
```

```
triple_omic <- create_triple_omic(
  measurement_df, feature_df, sample_df,
  "feature_id", "sample_id"
)
```

`downsample_heatmap` *Downsample Heatmap*

Description

Combine rows to speed up rendering of large heatmaps

Usage

```
downsample_heatmap(
  tidy_data,
  value_var,
  design,
  max_display_features = 1000,
  verbose = TRUE
)
```

Arguments

`tidy_data`	The data frame from a `tidy_omic` object containing ordered feature and sample primary keys defined by ordered_featureId and ordered_sampleId.
`value_var`	which variable in "measurements" to use for quantification.
`design`	a list summarizing the design of the tidy dataset
`max_display_features`	aggregate and downsample distinct feature to this number to speed to up heatmap rendering.
`verbose`	extra reporting messages

Value

tidy_data with rows collapsed if the number of distinct features is greater than `max_display_features`

export_tomic_as_tidy *Export T*Omic in Tidy Format*

Description

Export a data table including all fields from features, samples and measurements.

Usage

```
export_tomic_as_tidy(tomic, dir_path, name_preamble, verbose = TRUE)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
dir_path	path to save outputs
name_preamble	start of output file name
verbose	extra reporting messages

Value

Export one table which is one row per peak, which includes all feature and sample attributes.

Examples

```
if (interactive()) {
  export_tomic_as_tidy(brauer_2008_triple, "/tmp", "brauer")
}
```

export_tomic_as_triple *Export T*Omic as Triple*

Description

Export features, samples and measurements tables

Usage

```
export_tomic_as_triple(tomic, dir_path, name_preamble, verbose = TRUE)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
dir_path	path to save outputs
name_preamble	start of output file name
verbose	extra reporting messages

Value

Export three tables:

- features: one row per features measured (i.e., a metabolite)
- sample: one row per sample
- measurements: one row per measurement (i.e., one metabolite in one sample)

Examples

```
if (interactive()) {
  export_tomic_as_triple(brauer_2008_triple, "/tmp", "brauer")
}
```

---

export_tomic_as_wide *Export T*Omic as Wide Data*

Description

abundances form a matrix with metabolites as rows and samples as columns. Use transpose to treat samples as rows filename

Usage

```
export_tomic_as_wide(
  tomic,
  dir_path,
  name_preamble,
  value_var = NULL,
  transpose = FALSE,
  verbose = TRUE
)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
dir_path	path to save outputs
name_preamble	start of output file name
value_var	measurement variable to use for the matrix
transpose	if TRUE then samples will be stored as rows. If FALSE (default) then samples will be columns.
verbose	extra reporting messages

Value

Export one table which contains metabolites as rows and samples as columns.

Examples

```
if (interactive()) {
  export_tomic_as_wide(brauer_2008_triple, "/tmp", "brauer")
}
```

`filterInput` *Filter Input*

Description

UI components for the filter module.

Usage

```
filterInput(id, filter_table)
```

Arguments

`id`	An ID string that corresponds with the ID used to call the module's UI function.
`filter_table`	table to filter

Value

A `shiny` UI

`filterServer` *Filter Server*

Description

Server components for the filter module.

Usage

```
filterServer(id, tidy_omic, filter_table)
```

Arguments

`id`	An ID string that corresponds with the ID used to call the module's UI function.
`tidy_omic`	an object of class tidy_omic produced by `create_tidy_omic`
`filter_table`	table to filter

Value

A `tidy_omic` with some features and/or samples filtered.

filter_tomic *Filter T* Omics*

Description

Filter a tidy or triple omic to entries of interest.

Usage

```
filter_tomic(
  tomic,
  filter_type,
  filter_table,
  filter_value,
  filter_variable = NULL,
  invert = FALSE
)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
filter_type	**category** filter filter_variable to categories specified in filter_value
	range filter filter_variable to using the range (i.e., lower and upper limit) provided in filter_value
	quo a quosure as a filter_value to a table of interest
filter_table	table where the filter should be applied
filter_value	values to filter based on
filter_variable	variable to apply the filter to
invert	If FALSE (default) entities will be retained; if TRUE, they will be removed.

Value

A tomic object where a subset of features, samples or measurmenets have been filtered.

Examples

```
filter_tomic(
  brauer_2008_triple,
  filter_type = "category",
  filter_table = "features",
  filter_variable = "BP",
  filter_value = c("biological process unknown", "vacuolar acidification")
)

filter_tomic(
  brauer_2008_triple,
```

```
  filter_type = "category",
  filter_table = "samples",
  filter_variable = "DR",
  filter_value = 0.05
)

filter_tomic(
  brauer_2008_tidy,
  filter_type = "range",
  filter_table = "samples",
  filter_variable = "DR",
  filter_value = c(0, 0.2)
)

filter_tomic(
  brauer_2008_triple,
  filter_type = "quo",
  filter_table = "features",
  filter_value = rlang::quo(BP == "biological process unknown")
)
```

`format_names_for_plotting` *Format Names for Plotting*

Description

Wrap long names over multiple lines so that they will look better on plots.

Usage

```
format_names_for_plotting(chars, width = 40, truncate_at = 80)
```

Arguments

`chars`	a character vector (or a variable that can be converted to one)
`width`	Positive integer giving target line width (in number of characters). A width less than or equal to 1 will put each word on its own line.
`truncate_at`	max character length

Value

a reformatted character vector of the same length as the input.

Examples

```
chars <- "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer
  ac arcu semper erat porttitor egestas. Etiam sagittis, sapien at mattis."

format_names_for_plotting(chars)
```

get_design_tbl *Get Design Table*

Description

Get a tabular summary of all variables.

Usage

```
get_design_tbl(tomic_or_design)
```

Arguments

tomic_or_design	Either a tomic object or its embedded design list

Value

a tibble reflecting the tomic object's design.

Examples

```
get_design_tbl(brauer_2008_triple)
get_design_tbl(brauer_2008_triple$design)
```

get_tomic_table *Get Tomic Table*

Description

Extract one of the specific tables from a tomic object

Usage

```
get_tomic_table(tomic, table_type)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
table_type	The type of table to extract from the tomic object.
	tidy one row per measurements with feature and sample attributes added. Equivalent to the $data field of a tidy omic object
	measurements one row per measurements defined a feature and sample foreign key. Equivalent to the $measurements field of a triple omic object

features one row per feature defined by a feature primary key. Equivalent to the $features field of a triple omic object

samples one row per sample defined by a sample primary key. Equivalent to the $samples field of a triple omic object

Value

a tibble matching the `table_type` of the `tomic` object

Examples

```
get_tomic_table(brauer_2008_triple, "samples")
get_tomic_table(brauer_2008_tidy, "features")
```

`ggBivOutput` *ggBivariate Output*

Description

UI components for the ggBivariate module.

Usage

```
ggBivOutput(id, return_brushed_points = FALSE)
```

Arguments

`id`	An ID string that corresponds with the ID used to call the module's UI function.
`return_brushed_points`	Return values selected on the plot

Value

A `shiny` UI

ggBivServer *ggBivariate Server*

Description

Server components for the ggBivariate module.

Usage

```
ggBivServer(id, tomic, plot_table, return_brushed_points = FALSE)
```

Arguments

id	An ID string that corresponds with the ID used to call the module's UI function.
tomic	Either a tidy_omic or triple_omic object
plot_table	table containing the data to be plotted
return_brushed_points	Return values selected on the plot

Value

a tomic_table if return_brushed_points is TRUE, and 0 otherwise

ggplotOutput *ggplot Output*

Description

UI components for the ggplot module.

Usage

```
ggplotOutput(
  id,
  default_data_type = "samples",
  default_plot_type = "univariate"
)
```

Arguments

id	An ID string that corresponds with the ID used to call the module's UI function.
default_data_type	Default data type selection
default_plot_type	Default plot type selection

Value

A shiny UI

---

ggplotServer *ggplot Server*

Description

Server components for the ggplot module.

Usage

```
ggplotServer(id, tomic, return_brushed_points = FALSE)
```

Arguments

id	An ID string that corresponds with the ID used to call the module's UI function.
tomic	Either a tidy_omic or triple_omic object
return_brushed_points	Return values selected on the plot

Value

a tibble of selected observations if return_brushed_points is TRUE. Otherwise, returns NULL.

---

ggUnivOutput *ggUnivariate Output*

Description

UI components for the ggUnivariate module.

Usage

```
ggUnivOutput(id, return_brushed_points = FALSE)
```

Arguments

id	An ID string that corresponds with the ID used to call the module's UI function.
return_brushed_points	Return values selected on the plot

Value

A shiny UI

ggUnivServer *ggUnivariate Server*

Description

Server components for the ggUnivariate module

Usage

```
ggUnivServer(id, tomic, plot_table, return_brushed_points = FALSE)
```

Arguments

id	An ID string that corresponds with the ID used to call the module's UI function.
tomic	Either a tidy_omic or triple_omic object
plot_table	table containing the data to be plotted
return_brushed_points	Return values selected on the plot

Value

a tomic_table if return_brushed_points is TRUE, and 0 otherwise.

hclust_order *Hierarchical clustering order*

Description

Format and hierarchically cluster a data.frame. If hclust could not normally be produced (usually because no samples are in common for a feature) pad the matrix with zeros and still calculate the distance

Usage

```
hclust_order(
  df,
  feature_pk,
  sample_pk,
  value_var,
  cluster_dim,
  distance_measure = "dist",
  hclust_method = "ward.D2"
)
```

Arguments

df	data.frame to cluster
feature_pk	variable uniquely defining a row
sample_pk	variable uniquely defining a sample
value_var	An abundance value to use with hclust
cluster_dim	rows, columns, or both
distance_measure	variable to use for computing dis-similarity **corr** pearson correlation **dist** euclidean distance
hclust_method	method from stats::hclust to use for clustering

Value

a list containing a hierarchically clustered set of rows and/or columns

Examples

```
library(dplyr)

df <- tidyr::crossing(letters = LETTERS, numbers = 1:10) %>%
  mutate(noise = rnorm(n()))
hclust_order(df, "letters", "numbers", "noise", "rows")
```

---

impute_missing_values *Impute Missing Values*

Description

Impute missing values using K-nearest neighbors imputation

Usage

```
impute_missing_values(
  tomic,
  impute_var_name = "imputed",
  value_var = NULL,
  ...
)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
impute_var_name	variable to create for imputed measurements
value_var	An abundance value to use with hclust
...	additional arguments to pass to impute.knn

Value

A tomic object with imputed measurements.

Examples

```
## Not run:
# Requires the 'impute' package from Bioconductor
impute_missing_values(brauer_2008_triple)

## End(Not run)
```

---

infer_tomic_table_type *Infer Tomic Table Type*

Description

From a tomic_table, choose whether it reflects features, samples or measurements

Usage

```
infer_tomic_table_type(tomic, tomic_table)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
tomic_table	A table taken from a tidy (i.e., augmented measurements) or triple omic dataset

Value

features, samples or measurements

---

lassoInput *Lasso Input*

Description

UI components for the lasso module.

Usage

```
lassoInput(id)
```

Arguments

id	An ID string that corresponds with the ID used to call the module's UI function.

Value

A shiny UI

---

lassoServer *Lasso Server*

Description

Take a subset of entries from a tomic table (generally selected using the lasso function) and then either filter a tomic object to these entries or tag the entries of interest with a user-specified variable.

Usage

```
lassoServer(id, tomic, tomic_table)
```

Arguments

id	An ID string that corresponds with the ID used to call the module's UI function.
tomic	Either a tidy_omic or triple_omic object
tomic_table	A table taken from a tidy (i.e., augmented measurements) or triple omic dataset

Value

A tomic object amended based on the lasso selection.

---

organizeInput *Organize Input*

Description

UI components for the organize input module.

Usage

```
organizeInput(id)
```

Arguments

id	An ID string that corresponds with the ID used to call the module's UI function.

Value

A shiny UI

organizeServer *Organize Servers*

Description

Server components for the organize input module.

Usage

```
organizeServer(id, tidy_omic, feature_vars, sample_vars, value_var)
```

Arguments

id	An ID string that corresponds with the ID used to call the module's UI function.
tidy_omic	an object of class tidy_omic produced by create_tidy_omic
feature_vars	variables available for arranging features
sample_vars	variables available for arrange samples
value_var	An abundance value to use with hclust

Value

A tomic with sorted features and/or samples.

plotsaverInput *Plot Saver Input*

Description

UI components for the plot saver module.

Usage

```
plotsaverInput(id, ui_format = "tall")
```

Arguments

id	An ID string that corresponds with the ID used to call the module's UI function.
ui_format	Set UI appearance **tall** stack all UI elements **wide** UI elements are side-by-side

Value

a shiny UI

plotsaverServer *Plot Saver Server*

Description

Server components for the plot saver module.

Usage

```
plotsaverServer(id, grob, filename = "grob.png")
```

Arguments

id	An ID string that corresponds with the ID used to call the module's UI function.
grob	a ggplot2 plot
filename	filename for saving plot. The extension will be respected by ggsave.

Value

None

plot_bivariate *Bivariate Plot*

Description

Create a scatter or boxplot from a tomic dataset.

Usage

```
plot_bivariate(
  tomic_table,
  x_var,
  y_var,
  color_var = NULL,
  shape_var = NULL,
  alpha_var = NULL,
  size_var = NULL
)
```

Arguments

tomic_table	A table taken from a tidy (i.e., augmented measurements) or triple omic dataset
x_var	x-axis variable
y_var	y-axis variable
color_var	coloring variable (NULL to suppress coloring)
shape_var	shape variable (NULL to suppress shape)
alpha_var	alpha variable or numeric for constant alpha (NULL to suppress alpha)
size_var	size variable or integer/numeric for constant size (NULL to suppress size)

Value

a ggplot2 grob

Examples

```
library(dplyr)

brauer_augmented <- brauer_2008_tidy %>%
  add_pcs(npcs = 5) %>%
  tomic_to("triple_omic")

tomic_table <- brauer_augmented$samples
plot_bivariate(tomic_table, "PC1", "PC2", "nutrient", "nutrient", 0.5, 10)
plot_bivariate(tomic_table, "PC1", "PC2", NULL)
plot_bivariate(tomic_table, "nutrient", "PC2", "nutrient")
```

---

plot_heatmap *Plot Heatmap*

Description

Generate a heatmap visualization of a features x samples matrix of measurements.

Usage

```
plot_heatmap(
  tomic,
  feature_var = NULL,
  sample_var = NULL,
  value_var = NULL,
  cluster_dim = "both",
  distance_measure = "dist",
  hclust_method = "ward.D2",
  change_threshold = Inf,
  max_display_features = 800,
  x_title = NULL,
```

```
  y_title = NULL,
  colorbar_title = NULL,
  transpose = FALSE
)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
feature_var	variable from "features" to use as a unique feature label.
sample_var	variable from "samples" to use as a unique sample label.
value_var	which variable in "measurements" to use for quantification.
cluster_dim	rows, columns, or both
distance_measure	variable to use for computing dis-similarity **corr** pearson correlation **dist** euclidean distance
hclust_method	method from stats::hclust to use for clustering
change_threshold	values with a more extreme absolute change will be thresholded to this value.
max_display_features	aggregate and downsample distinct feature to this number to speed to up heatmap rendering.
x_title	label for x-axis (if NULL then use feature_var)
y_title	label for y-axis (if NULL then use sample_var)
colorbar_title	label for color-bar; default is log2 abundance
transpose	if TRUE then samples will be rows and features will be columns. Set all other variables as if transpose was FALSE.

Value

a ggplot2 grob

Examples

```
library(dplyr)

tomic <- brauer_2008_triple %>%
  filter_tomic(
    filter_type = "category",
    filter_table = "features",
    filter_variable = "BP",
    filter_value = c(
      "protein biosynthesis",
      "rRNA processing", "response to stress"
    )
  )
```

```
plot_heatmap(
  tomic = tomic,
  value_var = "expression",
  change_threshold = 5,
  cluster_dim = "rows",
  distance_measure = "corr",
  transpose = FALSE
)
```

`plot_missing_values` *Plot Missing Values*

Description

Create a simple plot of missing values.

Usage

```
plot_missing_values(tomic, value_var = NULL)
```

Arguments

`tomic`	Either a `tidy_omic` or `triple_omic` object
`value_var`	the measurement variable to check for missingness (NA or no entry)

Value

a ggplot2 grob

Examples

```
plot_missing_values(brauer_2008_triple)
```

`plot_univariate` *Univariate Plot*

Description

Create a histogram from a tomic dataset.

Usage

```
plot_univariate(tomic_table, x_var, color_var = NULL)
```

Arguments

tomic_table	A table taken from a tidy (i.e., augmented measurements) or triple omic dataset
x_var	x-axis variable
color_var	coloring variable (NULL to suppress coloring)

Value

A ggplot2 grob

Examples

```
library(dplyr)

brauer_augmented <- brauer_2008_tidy %>%
  add_pcs(npcs = 5) %>%
  tomic_to("triple_omic")

plot_univariate(brauer_augmented$samples, "PC1", "nutrient")
plot_univariate(brauer_augmented$measurements, "expression", NULL)
```

---

prepare_example_datasets

Prepare Example Datasets

Description

Format example datasets and add them to the package.

Usage

```
prepare_example_datasets(seed = 1234)
```

Arguments

seed	a seed value used to reproducibly sample random genes.

Value

None; used for side-effects.

reconcile_triple_omic *Reconcile Triple Omic*

Description

If some samples, feature or measurements have been dropped; update other tables.

Usage

```
reconcile_triple_omic(triple_omic)
```

Arguments

triple_omic	an object of class triple_omic produced by create_triple_omic

Value

a triple_omic object

reform_tidy_omic *Reform Tidy Omic*

Description

This function recreates a ‘tidy_omic‘ object from the "data" and "design" attributes of this object.

Usage

```
reform_tidy_omic(tidy_data, tomic_design)
```

Arguments

tidy_data	A tibble containing measurements along with sample metadata. This table can be obtained as the "data" attribute from a romic "tidy_omic" object.
tomic_design	a list with named attributes describing feature, sample, and measurement variables.

Details

This is handy for passing data and metadata through approaches like parsnip which expect data to be formatted as a data.frame

Examples

```
tidy_data <- romic::brauer_2008_tidy$data
reform_tidy_omic(tidy_data, romic::brauer_2008_tidy$design)
```

`remove_missing_values` *Remove Missing Values*

Description

Account for missing values by dropping features, samples or using imputation.

Usage

```
remove_missing_values(
  tomic,
  value_var = NULL,
  missing_val_method = "drop_samples",
  missing_value_types = c("NA", "NaN", "Inf"),
  verbose = FALSE
)
```

Arguments

`tomic`	Either a `tidy_omic` or `triple_omic` object
`value_var`	An abundance value to use with `hclust`
`missing_val_method`	Approach to remove missing values: **drop_features** Drop features with missing values **drop_samples** Drop samples which are missing all features, then drop features **impute** Impute missing values
`missing_value_types`	Types of values to treat as missing. One or more of "NA", "NaN" and "Inf"
`verbose`	extra reporting messages

Value

A `tomic` object where missing values have been accounted for.

Examples

```
remove_missing_values(brauer_2008_triple)
```

shiny_filter_test *Shiny Filter Test*

Description

Tests the shiny filter module as a stand-alone application.

Usage

```
shiny_filter_test(tidy_omic, filter_table = "features")
```

Arguments

tidy_omic	an object of class tidy_omic produced by create_tidy_omic
filter_table	table to filter

Value

A shiny app

Examples

```
if (interactive()) {
  shiny_filter_test(brauer_2008_tidy)
}
```

shiny_ggbiv_test *Shiny ggBivariate Test*

Description

Test the shiny ggBivariate module as a stand-alone application.

Usage

```
shiny_ggbiv_test(tomic, plot_table = "samples")
```

Arguments

tomic	Either a tidy_omic or triple_omic object
plot_table	table containing the data to be plotted

Value

a shiny app

Examples

```
if (interactive()) {
  shiny_ggbiv_test(
    add_pcs(brauer_2008_triple, npcs = 5),
    plot_table = "samples"
  )
  shiny_ggbiv_test(
    brauer_2008_triple,
    plot_table = "measurements"
  )
}
```

`shiny_ggplot_test` *Shiny ggplot Test*

Description

Test the shiny ggplot module as a stand-alone application.

Usage

```
shiny_ggplot_test(tomic)
```

Arguments

`tomic`	Either a `tidy_omic` or `triple_omic` object

Value

A `shiny` app

Examples

```
if (interactive()) {
  shiny_ggplot_test(add_pcs(brauer_2008_triple, npcs = 5))
  shiny_ggplot_test(brauer_2008_triple)
}
```

shiny_gguniv_test *Shiny ggUnivariate Test*

Description

Test the shiny ggUnivariate module as a stand-alone application.

Usage

```
shiny_gguniv_test(tomic, plot_table = "samples")
```

Arguments

tomic	Either a tidy_omic or triple_omic object
plot_table	table containing the data to be plotted

Value

A shiny app

Examples

```
if (interactive()) {
  shiny_gguniv_test(
    add_pcs(brauer_2008_triple, npcs = 5),
    plot_table = "samples"
  )
  shiny_gguniv_test(brauer_2008_triple, plot_table = "measurements")
  shiny_gguniv_test(brauer_2008_triple, plot_table = "features")
}
```

shiny_lasso_test *Shiny Lasso Test*

Description

Tests the shiny lasso module as a stand-alone application.

Usage

```
shiny_lasso_test(tomic, tomic_table)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
tomic_table	A table taken from a tidy (i.e., augmented measurements) or triple omic dataset

Value

A shiny app

Examples

```
if (interactive()) {
  tomic <- brauer_2008_triple
  tomic_table <- tomic[["samples"]] %>% dplyr::filter(nutrient == "G")
  shiny_lasso_test(tomic, tomic_table)
}
```

---

shiny_lasso_test_reactval

Shiny Lasso Test w/ Reactive Values

Description

Tests the shiny lasso module as a stand-alone application when the tomic is a reativeVal.

Usage

```
shiny_lasso_test_reactval(tomic, tomic_table)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
tomic_table	A table taken from a tidy (i.e., augmented measurements) or triple omic dataset

Value

A shiny app

Examples

```
if (interactive()) {
  tomic <- brauer_2008_triple
  tomic_table <- tomic[["samples"]] %>% dplyr::filter(nutrient == "G")
  shiny_lasso_test_reactval(tomic, tomic_table)

  tomic_table <- tomic[["measurements"]] %>% dplyr::filter(expression < -3)
  shiny_lasso_test_reactval(tomic, tomic_table)
}
```

shiny_organize_test *Shiny Organize Test*

Description

Tests the shiny organization module as stand-alone application.

Usage

```
shiny_organize_test(tidy_omic, feature_vars, sample_vars, value_var)
```

Arguments

tidy_omic	an object of class tidy_omic produced by create_tidy_omic
feature_vars	variables available for arranging features
sample_vars	variables available for arrange samples
value_var	An abundance value to use with hclust

Value

a shiny app

Examples

```
if (interactive()) {
  shiny_organize_test(
    brauer_2008_tidy,
    feature_vars = c("BP", "MF"),
    sample_vars = c("sample", "nutrient", "DR"),
    value_var = "expression"
  )
}
```

shiny_plotsaver_test *Shiny Plot Saver Test*

Description

Test the shiny plotsaver module as a stand-alone application.

Usage

```
shiny_plotsaver_test()
```

Value

a shiny app

Examples

```
if (interactive()) {
  shiny_plotsaver_test()
}
```

`shiny_sort_test` *Shiny Sort Test*

Description

Test the shiny sorting module as a stand-alone app.

Usage

```
shiny_sort_test(triple_omic, valid_sort_vars, value_var)
```

Arguments

`triple_omic`	an object of class triple_omic produced by `create_triple_omic`
`valid_sort_vars`	variables available for categorical arranging
`value_var`	An abundance value to use with `hclust`

Value

a shiny app

Examples

```
if (interactive()) {
  shiny_sort_test(brauer_2008_triple,
    valid_sort_vars = c("sample", "nutrient", "DR"),
    value_var = "expression"
  )
}
```

sortInput *Sort Input*

Description

UI components for the sort module.

Usage

```
sortInput(id, sort_table)
```

Arguments

id	An ID string that corresponds with the ID used to call the module's UI function.
sort_table	table to sort

Value

A shiny UI

sortServer *Sort Server*

Description

Server components for the sort module.

Usage

```
sortServer(id, tomic, sort_table, valid_sort_vars = NULL, value_var = NULL)
```

Arguments

id	An ID string that corresponds with the ID used to call the module's UI function.
tomic	Either a tidy_omic or triple_omic object
sort_table	samples or features
valid_sort_vars	variables available for categorical arranging
value_var	An abundance value to use with hclust

Value

A sorted tomic object.

`sort_tomic` *Sort Triple Omic*

Description

Sort a dataset's features or samples

Usage

```
sort_tomic(
  tomic,
  sort_type,
  sort_table,
  sort_variables = NULL,
  value_var = NULL
)
```

Arguments

`tomic`	Either a `tidy_omic` or `triple_omic` object
`sort_type`	**hclust** Arrange samples by hierarchical clustering of a provided `value_var` **arrange** Arrange samples by the factor or alphanumeric ordering of a set of `sort_variables`
`sort_table`	samples or features
`sort_variables`	A set of attributes in sort_table to sort with in `arrange`.
`value_var`	An abundance value to use with `hclust`

Details

`sort_tomic` supports the reordering of features or samples using either hierarchical clustering or based on the levels of other variables. Sorting occurs by turning either the feature or sample primary key into a factor whose levels reflect the sort.

Value

A `tomic` object where feature or sample primary keys have been turned into a factor reflecting how they are sorted.

Examples

```
library(dplyr)

sort_tomic(brauer_2008_triple,
  sort_type = "arrange", sort_table = "samples",
  sort_variables = c("nutrient", "DR")
) %>%
  sort_tomic(
```

```
  sort_type = "hclust",
  sort_table = "features",
  value_var = "expression"
)
```

`sort_triple_arrange` *Sort Triple Arrange*

Description

Sort a `triple_omic` object based on the values of one or more variables.

Usage

```
sort_triple_arrange(triple_omic, sort_table, sort_variables)
```

Arguments

`triple_omic`	an object of class triple_omic produced by `create_triple_omic`
`sort_table`	samples or features
`sort_variables`	A set of attributes in sort_table to sort with in `arrange`.

Value

A `triple_omic` with sorted features or samples.

`sort_triple_hclust` *Sort Triple Hclust*

Description

Sort a `triple_omic` object using hierarchical clustering

Usage

```
sort_triple_hclust(triple_omic, sort_table, value_var)
```

Arguments

`triple_omic`	an object of class triple_omic produced by `create_triple_omic`
`sort_table`	samples or features
`value_var`	An abundance value to use with `hclust`

Value

A `triple_omic` with clustered features or samples.

tidy_to_triple *Tidy omic to triple omic*

Description

Convert a `tidy_omic` object into a `triple_omic` object.

Usage

```
tidy_to_triple(tidy_omic)
```

Arguments

`tidy_omic`	an object of class tidy_omic produced by `create_tidy_omic`

Details

The `data` table will be converted into `features`, `samples`, and `measurements` tables using the `design` to determine which variables belong in each table. The `design` will be preserved as-is.

Value

A `triple_omic` object as created by `create_triple_omic`

Examples

```
tidy_to_triple(brauer_2008_tidy)
```

tomic_sort_status *T* Omic Sort Status*

Description

Determine whether features &/or samples have been sorted and stored as ordered_featureId and ordered_sampleId.

Usage

```
tomic_sort_status(tomic)
```

Arguments

`tomic`	Either a `tidy_omic` or `triple_omic` object

Value

length 1 character string indicating whether the `tomic` is sorted.

Examples

```
tomic_sort_status(brauer_2008_tidy)
```

---

tomic_to *T* Omic To*

Description

Takes in any romic reprsentation of a dataset and returns a specific representation.

Usage

```
tomic_to(tomic, to_class)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
to_class	The class to return, either tidy_omic or triple_omic

Value

tomic transformed to to_class class (or un-transformed if it started that way).

Examples

```
tomic_to(brauer_2008_tidy, "triple_omic")
```

---

tomic_to_matrix *Tomic To Matrix*

Description

Convert a T*Omic object to a feature x sample matrix matching the feature and sample ordering of a Triple Omic object.

Usage

```
tomic_to_matrix(tomic, value_var = NULL, transpose = FALSE)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
value_var	measurement variable to use for the matrix
transpose	if TRUE then samples will be stored as rows. If FALSE (default) then samples will be columns.

Details

Comparing the matrix to feature or sample variable vectors should work because the orders are matched. But, if features or samples are reordered after creating the matrix then the matrix's dimensions will no longer be aligned to feature and samples.

Value

a matrix with features as rows and samples as columns (if transpose FALSE) or features as columns and samples as rows (if transpose is TRUE).

Examples

```
tomic_to_matrix(brauer_2008_triple)
```

---

triple_to_tidy *Triple Omic to Tidy Omic*

Description

Convert a `triple_omic` object into a `tidy_omic` oobject.

Usage

```
triple_to_tidy(triple_omic)
```

Arguments

`triple_omic`	an object of class triple_omic produced by `create_triple_omic`

Details

Features, samples and measurements will be merged into a single `data` table, and the `design` will be preserved as-is.

Value

A `tidy_omic` object as created by `create_tidy_omic`.

Examples

```
library(dplyr)

measurement_df <- tidyr::expand_grid(
  feature_id = 1:10,
  sample_id = LETTERS[1:5]
) %>%
  dplyr::mutate(value = rnorm(n()))

feature_df <- tibble(
```

```
  feature_id = 1:10,
  feature_group = rep(c("a", "b"), each = 5)
)
sample_df <- tibble(
  sample_id = LETTERS[1:5],
  sample_group = c("a", "a", "b", "b", "b")
)

triple_omic <- create_triple_omic(
  measurement_df, feature_df, sample_df,
  "feature_id", "sample_id"
)
triple_to_tidy(triple_omic)
```

`try_brushedPoints` *Try brushedPoints*

Description

This function wraps brushedPoints in a try statement to catch cases where the brushing is out-of-sync with the df that is selected.

Usage

```
try_brushedPoints(...)
```

Arguments

`...`	args to pass to brushedPoints

Value

a df of brushed points

`update_sample_factors` *Update Sample Factors*

Description

Update sample metadata to order categorical variables based on a specified factor order.

Usage

```
update_sample_factors(tomic, factor_levels)
```

Arguments

tomic	Either a tidy_omic or triple_omic object
factor_levels	a character vector specifying the ordering of factor levels.

Value

a tomic object with updated sample metadata

Examples

```
update_sample_factors(
  brauer_2008_tidy, list(nutrient = c("G", "N", "P", "S", "L", "U"))
)
```

---

update_tidy_omic *Update Tidy Omic*

Description

Update a Tidy 'Omics data and schema to reflect newly added fields.

Usage

```
update_tidy_omic(tidy_omic, updated_tidy_data, new_variable_tables = c())
```

Arguments

tidy_omic	an object of class tidy_omic produced by create_tidy_omic
updated_tidy_data	a tibble of data to use to update tidy_omic.
new_variable_tables	a named character vector of newly added variables in updated_tidy_data (names) and the table (features, samples, measurements) they apply to (values).

Value

a tidy_omic object with an updated schema and/or data.

Examples

```
library(dplyr)

tidy_omic <- brauer_2008_tidy
updated_tidy_data <- tidy_omic$data %>%
  mutate(new_sample_var = "foo") %>%
  select(-DR)
new_variable_tables <- c("new_sample_var" = "samples")
```

```
update_tidy_omic(tidy_omic, updated_tidy_data, new_variable_tables)
```

`update_tomic` *Update T* Omic*

Description

Provide an updated features, samples or measurements table to a `tomic`.

Usage

```
update_tomic(tomic, tomic_table)
```

Arguments

`tomic`	Either a `tidy_omic` or `triple_omic` object
`tomic_table`	A table taken from a tidy (i.e., augmented measurements) or triple omic dataset

Value

A `tomic` object with updated features, samples or measurements.

Examples

```
library(dplyr)
updated_features <- brauer_2008_triple$features %>%
  dplyr::filter(BP == "biological process unknown") %>%
  dplyr::mutate(chromosome = purrr::map_int(systematic_name, function(x) {
    which(LETTERS == stringr::str_match(x, "Y([A-Z])")[2])
  }))

update_tomic(brauer_2008_triple, updated_features)
```

`var_partial_match` *Var Partial Match*

Description

Partial string matching of a provided variable to the variables available in a table

Usage

```
var_partial_match(x, df)
```

Arguments

x	a variable name or regex match to a variable name
df	a data.frame or tibble

Value

a single variable from df

Index