

Package: glystats (via r-universe)

October 16, 2026

Title Statistical Analysis of Glycoproteomics and Glycomics Data

Version 0.13.0

Description Provides a unified toolbox for bioinformatics analyses of glycomics and glycoproteomics data. Implemented methods include differential testing (independent and paired t-tests, Wilcoxon tests, one-way and repeated-measures ANOVA, Kruskal–Wallis and Friedman tests), formula-based moderated linear models, correlated-set Hotelling tests, multivariate modeling and visualization (PCA, t-SNE, UMAP), supervised learning (PLS-DA, OPLS-DA), clustering (k-means, hierarchical), survival modeling (Cox proportional hazards), correlation analysis, and ROC/AUC evaluation. All user-facing functions follow the gly_*() naming convention to facilitate auto-completion in RStudio. Analysis functions accept 'glyexp' 'GlycomiSE', 'GlycoproteomicSE', and 'SummarizedExperiment' objects as their unified data interface.

License MIT + file LICENSE

Suggests testthat (>= 3.0.0), FSA, emmeans, ropis, Rtsne, uwot, pROC, gg dendro, Hmisc, limma, knitr, rmarkdown, glyclean (>= 0.12.0), glyread (>= 0.8.4)

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL <https://glycoverse.github.io/glystats/>

Imports tibble, glyexp (>= 0.16.0), rlang, tidyr, dplyr, tidyselect, magrittr, janitor, checkmate, cli, stats, broom, purrr, stringr, survival, glyrepr (>= 0.9.0), SummarizedExperiment, S4Vectors, methods

Depends R (>= 4.1)

VignetteBuilder knitr

Config/pak/sysreqs libglpk-dev libicu-dev libxml2-dev zlib1g-dev

Repository <https://glycoverse.r-universe.dev>

Date/Publication 2026-10-16 03:43:40 UTC
RemoteUrl <https://github.com/glycoverse/glystats>
RemoteRef v0.13.0
RemoteSha 683d181aee566eb58caa90b46f83f3de7dbe75c6

Contents

filter_sig_vars	2
get_tidy_result	5
gly_ancova	6
gly_anova	8
gly_cor	10
gly_cox	12
gly_fold_change	13
gly_hclust	14
gly_kmeans	16
gly_kruskal	18
gly_limma	20
gly_linear_model	22
gly_oplsda	24
gly_pca	26
gly_plsda	27
gly_roc	29
gly_set_test	31
gly_tsne	33
gly_ttest	34
gly_umap	36
gly_wilcox	37
Index	40

filter_sig_vars	<i>Filter Significant Variables</i>
-----------------	-------------------------------------

Description

Filtering the experiment to keep only significant variables is a common task. This function provides a convenient way to do this. It supports results from all glystats DEA functions including [gly_anova\(\)](#), [gly_ancova\(\)](#), [gly_kruskal\(\)](#), [gly_ttest\(\)](#), [gly_wilcox\(\)](#), [gly_limma\(\)](#), and [gly_linear_model\(\)](#).

Usage

```
filter_sig_vars(  
  exp,  
  res,  
  p_adj_cutoff = 0.05,  
  p_val_cutoff = NULL,  
  fc_cutoff = NULL,  
  ...  
)  
  
## S3 method for class 'glystats_anova_res'  
filter_sig_vars(  
  exp,  
  res,  
  p_adj_cutoff = 0.05,  
  p_val_cutoff = NULL,  
  fc_cutoff = NULL,  
  on = "main_test",  
  comparison = NULL,  
  ...  
)  
  
## S3 method for class 'glystats_ancova_res'  
filter_sig_vars(  
  exp,  
  res,  
  p_adj_cutoff = 0.05,  
  p_val_cutoff = NULL,  
  fc_cutoff = NULL,  
  on = "main_test",  
  comparison = NULL,  
  ...  
)  
  
## S3 method for class 'glystats_kruskal_res'  
filter_sig_vars(  
  exp,  
  res,  
  p_adj_cutoff = 0.05,  
  p_val_cutoff = NULL,  
  fc_cutoff = NULL,  
  on = "main_test",  
  comparison = NULL,  
  ...  
)  
  
## S3 method for class 'glystats_ttest_res'  
filter_sig_vars(  
  exp,  
  res,  
  p_adj_cutoff = 0.05,  
  p_val_cutoff = NULL,  
  fc_cutoff = NULL,  
  on = "main_test",  
  comparison = NULL,  
  ...  
)
```

```

    exp,
    res,
    p_adj_cutoff = 0.05,
    p_val_cutoff = NULL,
    fc_cutoff = NULL,
    ...
)

## S3 method for class 'glystats_wilcox_res'
filter_sig_vars(
  exp,
  res,
  p_adj_cutoff = 0.05,
  p_val_cutoff = NULL,
  fc_cutoff = NULL,
  ...
)

## S3 method for class 'glystats_limma_res'
filter_sig_vars(
  exp,
  res,
  p_adj_cutoff = 0.05,
  p_val_cutoff = NULL,
  fc_cutoff = NULL,
  comparison = NULL,
  ...
)

## S3 method for class 'glystats_linear_model_res'
filter_sig_vars(
  exp,
  res,
  p_adj_cutoff = 0.05,
  p_val_cutoff = NULL,
  fc_cutoff = NULL,
  term = NULL,
  ...
)

```

Arguments

exp	A <code>glyexp::GlycomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another SummarizedExperiment. Use the same object used to generate the DEA result.
res	A glystats result object from a glystats DEA function.
p_adj_cutoff	The threshold for p-adjusted values. Default is 0.05.

p_val_cutoff	The threshold for p-values. We don't recommend using this. Default is NULL. If you insist to use it, please set p_adj_cutoff to NULL.
fc_cutoff	The threshold for fold changes. Only positive value is needed. For example, 2 means fold change > 2 or < 1/2. Default is 2 for glycoproteomics data and NULL for others.
...	Additional arguments passed to methods. See the method-specific documentation for details.
on	(For <code>gly_anova()</code> and <code>gly_kruskal()</code> results only) "main_test" or "post_hoc_test". Should the filter be applied on the main test results or the post-hoc test results? Default is "main_test". If "post_hoc_test", please set a comparison value.
comparison	(For <code>gly_anova()</code> , <code>gly_kruskal()</code> , and <code>gly_limma()</code> results only) Specifies which comparison to filter on. A string with the format "group1_vs_group2". For <code>gly_anova()</code> and <code>gly_kruskal()</code> results, comparison is only used when on is "post_hoc_test". If not provided, filtering will be performed on the main test results for <code>gly_anova()</code> and <code>gly_kruskal()</code> , and variables will be kept if they are significant in any comparison for <code>gly_limma()</code> .
term	(For <code>gly_linear_model()</code> results only) A coefficient or contrast name to filter on. If NULL, variables significant for any reported term are retained.

Value

A filtered object of the same class as exp.

Examples

```
library(glyexp)
library(glyclean)

exp <- auto_clean(real_experiment) |>
  glyexp::slice_head_row(n = 10)
res <- gly_anova(exp)
sig_exp <- filter_sig_vars(exp, res)
```

get_tidy_result	<i>Get the result of a glystats analysis</i>
-----------------	--

Description

Syntax sugar for `$tidy_result` and `$raw_result` elements of a glystats result object. It's useful to be used in pipes.

Usage

```
get_tidy_result(res, which = NULL)

get_raw_result(res, which = NULL)
```

Arguments

<code>res</code>	A glystats result object.
<code>which</code>	Used to specify which element to get, when the result is a list. For glystats results with only one tidy result (e.g. <code>gly_ttest()</code>), this argument is not needed. For others (e.g. <code>gly_anova()</code>), this argument is required to specify which tidy result to get.

Value

A tibble.

Examples

```
library(glyexp)
library(glyclean)
library(dplyr)

exp <- auto_clean(real_experiment) |>
  glyexp::slice_head_row(n = 10)

# Using a pipe
sig_res <- exp |>
  gly_ancova() |>
  get_tidy_result("main_test") |>
  filter(p_adj < 0.05)

# Equivalent to
anova_res <- gly_ancova(exp)
sig_res <- anova_res$tidy_result$main_test |>
  filter(p_adj < 0.05)
```

gly_ancova	<i>Analysis of Covariance (ANCOVA) for Differential Expression Analysis</i>
------------	---

Description

Perform ANCOVA for glycomics or glycoproteomics data with sample-level covariates. The function supports parametric comparison of multiple groups while adjusting for covariates. For significant results, emmeans post-hoc comparisons (Tukey adjustment) are automatically performed. P-values are adjusted for multiple testing using the method specified by `p_adj_method`.

Usage

```
gly_ancova(
  exp,
  group_col = "group",
```

```

covariate_cols = NULL,
p_adj_method = "BH",
add_info = TRUE,
...,
log_transform = TRUE
)

```

Arguments

exp	A <code>glyexp::GlycomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information.
group_col	A character string specifying the column name of the grouping variable in the sample information. Default is "group".
covariate_cols	A character vector specifying column names in sample information to include as covariates. At least one covariate must be provided.
p_adj_method	A character string specifying the method to adjust p-values. See <code>p.adjust.methods</code> for available methods. Default is "BH". If <code>NULL</code> , no adjustment is performed.
add_info	A logical value. If <code>TRUE</code> (default), variable information from the experiment will be added to the result tibble. If <code>FALSE</code> , only the statistical results are returned.
...	Additional arguments passed to <code>stats::aov()</code> .
log_transform	A single non-missing logical value. If <code>TRUE</code> (default), transform expression values using $\log_2(x + 1e-6)$ before analysis. If <code>FALSE</code> , use the supplied values unchanged, for example for already log-transformed data.

Details

Tests and effect sizes use the selected analysis scale. Reported fold changes continue to use ratios of group means from the original supplied values.

By default, expression data are transformed using $\log_2(x + 1e-6)$ before statistical testing. Set `log_transform = FALSE` to skip this transformation. At least 2 groups and at least 1 covariate are required.

For any variable failed to fit a `stats::aov()` model, NAs will be assigned to the results in both main test and post-hoc test.

Post-hoc Test: `emmeans` pairwise comparisons with Tukey adjustment (`emmeans::emmeans()`) are performed for variables with significant main effects (`p_adj < 0.05`).

Value

A list containing two elements:

- `tidy_result`: A list containing:
 - `main_test`: A tibble with ANCOVA results containing the following columns:
 - * `variable`: Variable name
 - * `term`: ANCOVA term (usually "group")

- * df: Degrees of freedom
- * sumsq: Sum of squares
- * meansq: Mean squares
- * statistic: F-statistic
- * p_val: Raw p-value from ANCOVA
- * p_adj: Adjusted p-value (if p_adj_method is not NULL)
- * post_hoc: Significant group pairs from post-hoc test, in the format of "ref_vs_test".
- post_hoc_test: A tibble with pairwise comparison results containing the following columns:
 - * variable: Variable name
 - * ref_group: Reference group
 - * test_group: Test/treatment/case group
 - * p_val: Adjusted p-value from emmeans pairwise test
 - * p_adj: Adjusted p-value from emmeans pairwise test
- raw_result: A list containing:
 - main_test: A list of raw aov model objects.
 - post_hoc_test: A list of raw emmeans results.

Required packages

This function requires the emmeans package for post-hoc comparisons.

See Also

`stats::aov()`, `emmeans::emmeans()`, `emmeans::contrast()`

gly_anova

One-way or Repeated-Measures ANOVA for Differential Expression Analysis

Description

Perform one-way or repeated-measures ANOVA for glycomics or glycoproteomics data. For significant results, Tukey's HSD post-hoc test is automatically performed. P-values are adjusted for multiple testing using the method specified by p_adj_method.

Usage

```
gly_anova(
  exp,
  group_col = "group",
  p_adj_method = "BH",
  add_info = TRUE,
  ...,
  subject_col = NULL,
  log_transform = TRUE
)
```

Arguments

exp	A <code>glyexp::GlycomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information.
group_col	A character string specifying the column name of the grouping variable in the sample information. Default is "group".
p_adj_method	A character string specifying the method to adjust p-values. See <code>p.adjust.methods</code> for available methods. Default is "BH". If NULL, no adjustment is performed.
add_info	A logical value. If TRUE (default), variable information from the experiment will be added to the result tibble. If FALSE, only the statistical results are returned.
...	Additional arguments passed to <code>stats::aov()</code> .
subject_col	An optional character string naming the subject identifier column in sample information. When supplied, a subject-blocked repeated-measures ANOVA is performed.
log_transform	A single non-missing logical value. If TRUE (default), transform expression values using $\log_2(x + 1e-6)$ before analysis. If FALSE, use the supplied values unchanged, for example for already log-transformed data.

Details

Tests and effect sizes use the selected analysis scale. Reported fold changes continue to use ratios of group means from the original supplied values.

By default, expression data are transformed using $\log_2(x + 1e-6)$ before statistical testing. Set `log_transform = FALSE` to skip this transformation. At least 2 groups are required in the grouping variable. In paired analyses, only subjects observed in every group are used, with at most one sample per subject and group. The reported effect size is partial eta-squared for the group term.

For any variable failed to fit a `stats::aov()` model, NAs will be assigned to the results in both main test and post-hoc test.

Post-hoc Test: Tukey's HSD test for pairwise comparisons (`stats::TukeyHSD()`) is performed for variables with significant main effects ($p_{adj} < 0.05$). Paired analyses instead use paired t-tests with Holm correction.

Value

A list containing three elements:

- `tidy_result`: A list containing:
 - `main_test`: A tibble with ANOVA results containing the following columns:
 - * `variable`: Variable name
 - * `term`: ANOVA term (usually "groups")
 - * `df`: Degrees of freedom
 - * `sumsq`: Sum of squares
 - * `meansq`: Mean squares
 - * `statistic`: F-statistic

- * p_val: Raw p-value from ANOVA
- * effect_size: Eta-squared, or partial eta-squared for paired analyses
- * p_adj: Adjusted p-value (if p_adj_method is not NULL)
- * post_hoc: Significant group pairs from post-hoc test, in the format of "ref_vs_test".
- post_hoc_test: A tibble with pairwise comparison results containing the following columns:
 - * variable: Variable name
 - * ref_group: Reference group
 - * test_group: Test/treatment/case group
 - * p_val: Raw post-hoc p-value
 - * p_adj: Adjusted post-hoc p-value
- raw_result: A list containing:
 - main_test: A list of raw aov model objects.
 - post_hoc_test: A list of raw TukeyHSD objects or paired t-test summaries.
- meta_data: A list containing metadata from the input experiment.

See Also

[stats::aov\(\)](#), [stats::TukeyHSD\(\)](#)

gly_cor

Correlation Analysis for Glycomics and Glycoproteomics Data

Description

Perform pairwise correlation analysis on variables or samples in the expression data. The function calculates correlation coefficients and p-values for all pairs, with optional multiple testing correction.

Usage

```
gly_cor(
  exp,
  on = "variable",
  method = "pearson",
  p_adj_method = "BH",
  ...,
  log_transform = TRUE
)
```

Arguments

exp	A <code>glyexp::GlycomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information.
on	A character string specifying what to correlate. Either "variable" (default) to correlate variables/features, or "sample" to correlate samples/observations.
method	A character string indicating which correlation coefficient is to be computed. One of "pearson" (default) or "spearman". Note: "kendall" is not supported by <code>Hmisc::rcorr</code> .
p_adj_method	A character string specifying the method to adjust p-values. See <code>p.adjust.methods</code> for available methods. Default is "BH". If NULL, no adjustment is performed.
...	Additional arguments passed to <code>Hmisc::rcorr()</code> .
log_transform	A single logical value indicating whether to transform expression values using $\log_2(x + 1e-6)$ before analysis. Default is TRUE. Set to FALSE for data that are already log-transformed.

Details

Set `log_transform = FALSE` to use the expression values as supplied, for example when they are already log-transformed.

By default, the function performs $\log_2$ transformation on the expression data ($\log_2(x + 1e-6)$) before correlation analysis. When `on = "variable"` (default), correlations are calculated between variables across samples. When `on = "sample"`, correlations are calculated between samples across variables.

Correlation Calculation: Correlation coefficients and p-values are calculated using `Hmisc::rcorr()` which is more efficient than `pairwise stats::cor.test()` calls.

Multiple Testing Correction: P-values are adjusted for multiple testing using the method specified by `p_adj_method`.

Value

A list with three elements:

- `tidy_result`: A tibble with correlation results containing the following columns:
 - `variable1` or `sample1`: First element of the pair (depending on `on` parameter)
 - `variable2` or `sample2`: Second element of the pair (depending on `on` parameter)
 - `cor`: Correlation coefficient
 - `p_val`: Raw p-value from correlation test
 - `p_adj`: Adjusted p-value (if `p_adj_method` is not NULL)
- `raw_result`: The raw `rcorr` object from `Hmisc::rcorr()`
- `meta_data`: A list containing metadata from the input experiment The list has classes `glystats_cor_res` and `glystats_res`.

Required packages

This function requires the `Hmisc` package for efficient correlation calculation.

See Also

`Hmisc::rcorr()`, `stats::cor()`

gly_cox

Cox Proportional Hazards Model for Survival Analysis

Description

Fit a Cox proportional hazards model for each variable in the expression data, and extract p-values and hazard ratios from it.

Usage

```
gly_cox(
  exp,
  time_col = "time",
  event_col = "event",
  p_adj_method = "BH",
  add_info = TRUE,
  ...,
  log_transform = FALSE
)
```

Arguments

exp	A <code>glyexp::GlycomiSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information.
time_col	A character string specifying the column name in sample information that contains survival time. Default is "time".
event_col	A character string specifying the column name in sample information that contains event indicator (1 for event, 0 for censoring). Default is "event".
p_adj_method	A character string specifying the method to adjust p-values. See <code>p.adjust.methods</code> for available methods. Default is "BH". If NULL, no adjustment is performed.
add_info	A logical value. If TRUE (default), variable information from the experiment will be added to the result tibble. If FALSE, only the Cox model results are returned.
...	Additional arguments passed to <code>survival::coxph()</code> .
log_transform	A single logical value indicating whether to transform expression values using $\log_2(x + 1e-6)$ before analysis. Default is FALSE. Set to FALSE for data that are already log-transformed.

Details

Expression values are used as supplied by default. Set `log_transform = TRUE` to apply $\log_2(x + 1e-6)$ before analysis.

The function fits an Cox proportional hazards model for each variable by:

```
coxph(Surv(time, event) ~ expression_value)
```

P-values are adjusted by Benjamini-Hochberg method by default.

Value

A list with three elements:

- `tidy_result`: A tibble with Cox model results containing the following columns:
 - `variable`: Variable name
 - `coefficient`: Regression coefficient (log hazard ratio)
 - `std.error`: Standard error of the coefficient
 - `statistic`: Wald test statistic
 - `p_val`: Raw p-value from Wald test
 - `hr`: Hazard ratio ($\exp(\text{coefficient})$)
 - `p_adj`: Adjusted p-value (if `p_adj_method` is not NULL)
- `raw_result`: A list of raw `coxph` model objects.
- `meta_data`: A list containing metadata from the input experiment

See Also

[survival::coxph\(\)](#), [survival::Surv\(\)](#)

gly_fold_change	<i>Calculate fold change</i>
-----------------	------------------------------

Description

Calculate fold change for a given expression matrix and group information. It could only be used for 2-group analysis. When you run this function, you will see message about "Ref Group" and "Test Group". "Ref Group" is the reference group, and "Test Group" is the test/treatment/case group.

Usage

```
gly_fold_change(  
  exp,  
  group_col = "group",  
  add_info = TRUE,  
  log_transform = FALSE  
)
```

Arguments

exp	A <code>glyexp::GlycomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> .
group_col	The column name of the group information in the sample information.
add_info	A logical value. If <code>TRUE</code> (default), variable information from the experiment will be added to the result tibble. If <code>FALSE</code> , only the fold change results are returned.
log_transform	A single non-missing logical. If <code>FALSE</code> (default), return the $\log_2$ ratio of arithmetic group means as before. If <code>TRUE</code> , apply $\log_2(x + 1e-6)$ and return the difference of group means on that scale, equivalent to the $\log_2$ ratio of geometric means of $x + 1e-6$. This option transforms input expression; it does not toggle the $\log_2$ scale of the reported fold change.

Value

A list with three elements:

- `tidy_result`: A tibble with fold change results containing the following columns:
 - `variable`: Variable name
 - `log2fc`: $\log_2$ ratio of arithmetic group means by default, or difference of mean $\log_2$ -transformed values when `log_transform = TRUE`. If more than two groups, two additional columns `ref_group` and `test_group` will be added.
- `raw_result`: The raw result (same as `tidy_result` for this function)
- `meta_data`: A list containing metadata from the input experiment

gly_hclust

Hierarchical Clustering for Glycomics and Glycoproteomics Data

Description

Perform hierarchical clustering on the expression data. The function uses `stats::hclust()` to perform clustering and provides tidy results including cluster assignments, dendrogram data for plotting, and merge heights.

Usage

```
gly_hclust(
  exp,
  on = "variable",
  k_values = c(2, 3, 4, 5),
  scale = TRUE,
  add_info = TRUE,
  ...,
  log_transform = TRUE
)
```

Arguments

exp	A <code>glyexp::GlycomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information.
on	A character string specifying what to cluster. Either "variable" (default) to cluster variables/features, or "sample" to cluster samples/observations.
k_values	A numeric vector specifying the number of clusters to cut the tree into. Default is <code>c(2, 3, 4, 5)</code> . If <code>NULL</code> , no cluster assignments are returned.
scale	A logical indicating whether to scale the data before clustering. Default is <code>TRUE</code> .
add_info	A logical value. If <code>TRUE</code> (default), sample information from the experiment will be added to the result tibbles. If <code>FALSE</code> , only the clustering results are returned.
...	Additional arguments passed to <code>stats::dist()</code> and <code>stats::hclust()</code> . Note: if both functions need a method parameter, use <code>dist.method</code> for distance and <code>hclust.method</code> for clustering method.
log_transform	A single logical value indicating whether to transform expression values using $\log_2(x + 1e-6)$ before analysis. Default is <code>TRUE</code> . Set to <code>FALSE</code> for data that are already log-transformed.

Details

Set `log_transform = FALSE` to use the expression values as supplied, for example when they are already log-transformed.

By default, the function performs $\log_2$ transformation on the expression data ($\log_2(x + 1e-6)$) before clustering. When `on = "variable"` (default), variables are clustered based on their expression patterns across samples. When `on = "sample"`, samples are clustered based on their expression profiles across variables.

Distance Calculation: Distance is calculated using `stats::dist()` with the specified method.

Clustering Method: Hierarchical clustering is performed using `stats::hclust()` with the specified method.

Cluster Assignment: The dendrogram is cut at different heights to produce cluster assignments for the specified `k` values using `stats::cutree()`.

Value

A list containing:

- `tidy_result`: A list of tibbles with clustering results:
 - `clusters`: Cluster assignments containing the following columns:
 - * `variable` or `sample`: Variable or sample name (depending on `on` parameter)
 - * `cluster_k2`, `cluster_k3`, etc.: Cluster assignments for different `k` values
 - `dendrogram`: Dendrogram segment data for plotting (if `ggdendro` is available) containing:
 - * `x`, `y`, `xend`, `yend`: Segment coordinates for plotting
 - `heights`: Merge heights and steps containing the following columns:

- * merge_step: Step number in the clustering process
- * height: Height at which clusters are merged
- * n_clusters: Number of clusters remaining after this merge
- labels: Labels and their positions (if gg dendro is available) containing:
 - * x, y: Position coordinates
 - * label: Label text
- raw_result: The raw hclust object from `stats::hclust()`
- meta_data: A list containing metadata from the input experiment

Required packages

This function only uses base R packages and does not require additional dependencies. For enhanced dendrogram plotting capabilities, the `ggdendro` package is recommended but not required.

See Also

`stats::hclust()`, `stats::dist()`, `stats::cutree()`

gly_kmeans

K-means Clustering for Glycomics and Glycoproteomics Data

Description

Perform k-means clustering on the expression data. The function uses `stats::kmeans()` to perform clustering and provides tidy results with cluster assignments.

Usage

```
gly_kmeans(
  exp,
  on = "variable",
  centers = 3,
  scale = TRUE,
  add_info = TRUE,
  ...,
  log_transform = TRUE
)
```

Arguments

- | | |
|-----|--|
| exp | A <code>glyexp::GlycomicsSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information. |
| on | A character string specifying what to cluster. Either "variable" (default) to cluster variables/features, or "sample" to cluster samples/observations. |

centers	Either the number of clusters (integer) or a set of initial cluster centers. Default is 3.
scale	A logical indicating whether to scale the data before clustering. Default is TRUE.
add_info	A logical value. If TRUE (default), sample information from the experiment will be added to the result tibbles. If FALSE, only the clustering results are returned.
...	Additional arguments passed to <code>stats::kmeans()</code> .
log_transform	A single logical value indicating whether to transform expression values using $\log_2(x + 1e-6)$ before analysis. Default is TRUE. Set to FALSE for data that are already log-transformed.

Details

Set `log_transform = FALSE` to use the expression values as supplied, for example when they are already log-transformed.

By default, the function performs $\log_2$ transformation on the expression data ($\log_2(x + 1e-6)$) before clustering. When `on = "variable"` (default), variables are clustered based on their expression patterns across samples. When `on = "sample"`, samples are clustered based on their expression profiles across variables.

Data Preparation: Data is log2-transformed when `log_transform = TRUE` and optionally scaled before clustering.

Clustering Method: K-means clustering is performed using `stats::kmeans()` with the specified parameters.

Value

A list with three elements:

- `tidy_result`: A tibble with cluster assignments containing the following columns:
 - `variable` or `sample`: Variable or sample name (depending on `on` parameter)
 - `cluster`: Cluster assignment
- `raw_result`: The raw kmeans object from `stats::kmeans()`.
- `meta_data`: A list containing metadata from the input experiment

Required packages

This function only uses base R packages and does not require additional dependencies.

See Also

[stats::kmeans\(\)](#)

gly_kruskal

*Kruskal-Wallis or Friedman Test for Differential Expression Analysis***Description**

Perform Kruskal-Wallis or Friedman tests for glycomics or glycoproteomics data. For significant results, Dunn's post-hoc test is automatically performed. P-values are adjusted for multiple testing using the method specified by `p_adj_method`.

Usage

```
gly_kruskal(
  exp,
  group_col = "group",
  p_adj_method = "BH",
  add_info = TRUE,
  ...,
  subject_col = NULL,
  log_transform = TRUE
)
```

Arguments

<code>exp</code>	A <code>glyexp::GlycomicsSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information.
<code>group_col</code>	A character string specifying the column name of the grouping variable in the sample information. Default is "group".
<code>p_adj_method</code>	A character string specifying the method to adjust p-values. See <code>p.adjust.methods</code> for available methods. Default is "BH". If <code>NULL</code> , no adjustment is performed.
<code>add_info</code>	A logical value. If <code>TRUE</code> (default), variable information from the experiment will be added to the result tibble. If <code>FALSE</code> , only the statistical results are returned.
<code>...</code>	Additional arguments passed to <code>stats::kruskal.test()</code> or, for paired analyses, <code>stats::friedman.test()</code> .
<code>subject_col</code>	An optional character string naming the subject identifier column in sample information. When supplied, a Friedman test is performed.
<code>log_transform</code>	A single non-missing logical value. If <code>TRUE</code> (default), transform expression values using $\log_2(x + 1e-6)$ before analysis. If <code>FALSE</code> , use the supplied values unchanged, for example for already log-transformed data.

Details

Tests and effect sizes use the selected analysis scale. Reported fold changes continue to use ratios of group means from the original supplied values.

By default, expression data are transformed using $\log_2(x + 1e-6)$ before statistical testing. Set `log_transform = FALSE` to skip this transformation. At least 2 groups are required in the grouping variable. In paired analyses, only subjects observed in every group are used, with at most one sample per subject and group. The reported effect size is Kendall's W .

For any variable failed to fit a `stats::kruskal.test()` or `stats::friedman.test()` model, NAs will be assigned to the results in both main test and post-hoc test.

Post-hoc Test: Dunn's test with Holm correction for multiple comparisons (`FSA::dunnTest()`) is performed for variables with significant main effects ($p_{\text{adj}} < 0.05$). Paired analyses instead use paired Wilcoxon signed-rank tests with Holm correction.

Value

A list containing three elements:

- `tidy_result`: A list containing:
 - `main_test`: A tibble with Kruskal-Wallis or Friedman test results containing the following columns:
 - * `variable`: Variable name
 - * `statistic`: Kruskal-Wallis test statistic
 - * `p_val`: Raw p-value from Kruskal-Wallis test
 - * `parameter`: Degrees of freedom
 - * `method`: Statistical method used
 - * `effect_size`: Epsilon-squared, or Kendall's W for paired analyses
 - * `p_adj`: Adjusted p-value (if `p_adj_method` is not NULL)
 - * `post_hoc`: Significant group pairs from post-hoc test, in the format of "ref_vs_test".
 - `post_hoc_test`: A tibble with pairwise comparison results containing the following columns:
 - * `variable`: Variable name
 - * `ref_group`: Reference group
 - * `test_group`: Test/treatment/case group
 - * `p_val`: Raw post-hoc p-value
 - * `p_adj`: Holm-adjusted post-hoc p-value
- `raw_result`: A list containing:
 - `main_test`: A list of raw `kruskal.test` or `friedman.test` objects.
 - `post_hoc_test`: A list of raw Dunn or paired Wilcoxon summaries.
- `meta_data`: A list containing metadata from the input experiment.

Required packages

Unpaired analyses require the `FSA` package for Dunn's post-hoc test.

See Also

`stats::kruskal.test()`, `stats::friedman.test()`, `FSA::dunnTest()`

gly_limma

*Linear Models for Differential Expression Analysis using limma***Description**

Perform differential expression analysis using linear models with empirical Bayes moderation from the limma package. Supports both two-group and multi-group comparisons.

Usage

```
gly_limma(
  exp,
  group_col = "group",
  covariate_cols = NULL,
  subject_col = NULL,
  p_adj_method = "BH",
  ref_group = NULL,
  contrasts = NULL,
  add_info = TRUE,
  ...,
  log_transform = TRUE
)
```

Arguments

exp	A <code>glyexp::GlycomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another SummarizedExperiment containing expression data and sample information.
group_col	A character string specifying the column name in sample information that contains group labels. Default is "group".
covariate_cols	A character vector specifying column names in sample information to include as covariates in the limma model. Default is NULL.
subject_col	A character string specifying the column name in sample information that contains subject identifiers for paired comparisons. Default is NULL.
p_adj_method	A character string specifying the method for multiple testing correction. Must be one of the methods supported by <code>stats::p.adjust()</code> . Default is "BH" (Benjamini-Hochberg). Set to NULL to skip p-value adjustment.
ref_group	A character string specifying the reference group. If NULL (default), the first level of the group factor is used as the reference. Only used for two-group comparisons.
contrasts	A character vector specifying custom contrasts. If NULL (default), all pairwise comparisons are automatically generated, and the levels coming first in the factor will be used as the reference group. Supports two formats: "group1-group2" or "group1_vs_group2". Use the second format if group names contain hyphens. "group1" will be used as the reference group.

<code>add_info</code>	A logical value. If TRUE (default), variable information from the experiment will be added to the result tibble. If FALSE, only the statistical results are returned.
<code>...</code>	Additional arguments passed to <code>limma::lmFit()</code> .
<code>log_transform</code>	A single non-missing logical value. If TRUE (default), transform expression values using $\log_2(x + 1e-6)$ before analysis. If FALSE, use the supplied values unchanged, for example for already log-transformed data.

Details

By default, expression data are transformed using $\log_2(x + 1e-6)$ before statistical testing. Set `log_transform = FALSE` to skip this transformation. The analysis uses linear models with empirical Bayes moderation to improve statistical power, especially for small sample sizes.

The `log2fc` column always contains the fitted model contrast. When `log_transform = FALSE`, its units are those of the supplied expression data; it is a $\log_2$ fold change only when those values are already on a $\log_2$ scale.

For two or more groups, all pairwise comparisons are automatically generated (one contrast for two groups) and performed using contrast matrices unless custom contrasts are specified.

If covariates are provided, they are added as additional terms in the design matrix and are not part of the group contrasts.

If subjects are provided, they are included as a blocking factor in the design matrix using the formula $\sim 0 + \text{group} + \text{subject}$ (plus any covariates).

When specifying custom contrasts, use either "group1-group2" or "group1_vs_group2" format. If group names contain hyphens and you use the first format, an error will be raised suggesting to use the second format.

Value

A list with three elements:

- `tidy_result`: A tibble with limma results containing the following columns:
 - `variable`: Variable name
 - `log2fc`: Model contrast on the analysis scale ($\log_2$ fold change when expression values are on a $\log_2$ scale)
 - `AveExpr`: Average expression level
 - `t`: t-statistic
 - `p_val`: Raw p-value
 - `p_adj`: Adjusted p-value (if `p_adj_method` is not NULL)
 - `b`: B-statistic (log-odds of differential expression)
 - `ref_group`: Reference group
 - `test_group`: Test group
- `raw_result`: The raw limma fit object(s).
- `meta_data`: A list containing metadata from the input experiment.

Required packages

This function requires the following packages to be installed:

- limma for linear model fitting and empirical Bayes moderation

See Also

`limma::lmFit()`, `limma::eBayes()`, `limma::makeContrasts()`

gly_linear_model	<i>Formula-based Linear Models</i>
------------------	------------------------------------

Description

Fit a linear model to every variable in a glycomics or glycoproteomics experiment using limma's empirical Bayes moderation.

Usage

```
gly_linear_model(
  exp,
  formula,
  contrasts = NULL,
  p_adj_method = "BH",
  add_info = TRUE,
  ...,
  log_transform = TRUE
)
```

Arguments

exp	A <code>glyexp::GlycomicsSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another SummarizedExperiment containing expression data and sample information.
formula	A one-sided formula describing the model in terms of columns in the sample information, for example <code>~ treatment * time + batch</code> .
contrasts	An optional uniquely named character vector of limma-style contrast expressions. Use backticks around non-syntactic coefficient names, such as <code>`treatmentB:timeT2`</code> .
p_adj_method	A character string specifying the method used to adjust p-values within each coefficient or contrast. See <code>stats::p.adjust()</code> for available methods. If <code>NULL</code> , no adjustment is performed.
add_info	A logical value. If <code>TRUE</code> (default), variable information from the experiment is added to the tidy result.
...	Additional arguments passed to <code>limma::lmFit()</code> .
log_transform	A single non-missing logical value. If <code>TRUE</code> (default), transform expression values using $\log_2(x + 1e-6)$ before analysis. If <code>FALSE</code> , use the supplied values unchanged, for example for already log-transformed data.

Details

`gly_linear_model()` and `gly_limma()` both use limma under the hood, but serve complementary use cases. `gly_linear_model()` provides a general formula interface for specifying a wide range of analysis designs and custom contrasts. `gly_limma()` is a dedicated wrapper for common differential expression analysis tasks, with convenient group, covariate, subject, and pairwise-comparison arguments.

By default, expression values are transformed using $\log_2(x + 1e-6)$ before modeling. Set `log_transform = FALSE` to use the supplied values unchanged. Coefficient estimates and average expression are reported on the selected analysis scale. When contrasts is NULL, all non-intercept coefficients are tested. When contrasts are supplied, only the named contrasts are returned.

Character predictors are handled as factors by `stats::model.matrix()`. Factor reference levels therefore determine the meaning of coefficients. Set factor levels in the sample information before calling this function when a specific reference level is required.

P-values are adjusted independently across variables for each coefficient or contrast.

Value

A list with classes `glystats_linear_model_res` and `glystats_res` containing:

- `tidy_result`: A tibble with `variable`, `term`, `estimate`, `AveExpr`, `t`, `p_val`, optional `p_adj`, and `b` columns.
- `raw_result`: A list containing the moderated limma fit, design matrix, contrast matrix, and coefficient-name mapping.
- `meta_data`: Metadata copied from `exp`.

See Also

`gly_limma()`, `limma::lmFit()`, `limma::eBayes()`

Examples

```
exp <- glyexp::real_experiment
complete <- rowSums(is.finite(SummarizedExperiment::assay(exp))) == ncol(exp)
exp <- exp[which(complete)[seq_len(10)], ]
SummarizedExperiment::colData(exp)$time <- factor(
  rep(c("early", "late"), length.out = ncol(exp))
)

model_res <- gly_linear_model(exp, ~ group * time)

contrast_res <- gly_linear_model(
  exp,
  ~ group * time,
  contrasts = c(
    C_late_vs_H_late = "groupC + `groupC:timelate`"
  )
)
```

gly_oplsda

*Orthogonal Partial Least Squares Discriminant Analysis (OPLS-DA)***Description**

Perform orthogonal partial least squares discriminant analysis on the expression data. The function uses `ropls::opls()` to perform OPLS-DA and returns tidy results.

Usage

```
gly_oplsda(
  exp,
  group_col = "group",
  pred_i = 1,
  ortho_i = NA,
  scale = TRUE,
  add_info = TRUE,
  ...,
  log_transform = TRUE
)
```

Arguments

<code>exp</code>	A <code>glyexp::GlycomomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information.
<code>group_col</code>	A character string specifying the column name in sample information that contains group labels. Default is "group".
<code>pred_i</code>	An integer indicating the number of predictive components to include. Default is 1.
<code>ortho_i</code>	An integer indicating the number of orthogonal components to include. Default is NA (automatic).
<code>scale</code>	A logical indicating whether to scale the data. Default is TRUE.
<code>add_info</code>	A logical value. If TRUE (default), sample and variable information from the experiment will be added to the result tibbles. If FALSE, only the OPLS-DA results are returned.
<code>...</code>	Additional arguments passed to <code>ropls::opls()</code> .
<code>log_transform</code>	A single logical value indicating whether to transform expression values using $\log(x + 1)$ before analysis. Default is TRUE. Set to FALSE for data that are already log-transformed.

Details

By default, expression values are transformed using $\log(x + 1)$ before analysis. Set `log_transform = FALSE` to use the expression values as supplied, for example when they are already log-transformed.

Value

A list containing:

- tidy_result: A list of tibbles with OPLS-DA results:
 - samples: OPLS-DA scores for each sample containing the following columns:
 - * sample: Sample name
 - * group: Group assignment
 - * p1, p2, etc.: Predictive component scores
 - * o1, o2, etc.: Orthogonal component scores
 - variables: OPLS-DA loadings for each variable containing the following columns:
 - * variable: Variable name
 - * p1, p2, etc.: Predictive component loadings
 - * o1, o2, etc.: Orthogonal component loadings
 - * pcorr1, pcorr2, etc.: Correlation between each variable and the corresponding predictive component
 - variance: OPLS-DA explained variance containing the following columns:
 - * component: Component name (p1, o1, etc.)
 - * prop_var_explained: Proportion of variance explained by each component
 - * cumulative_prop_var: Cumulative proportion of variance explained
 - vip: Variable Importance in Projection scores containing the following columns:
 - * variable: Variable name
 - * vip: VIP score
 - perm_test: Permutation test results containing the following columns:
 - * model: Model type ("Original" for the original model, "Permutation" for permuted models)
 - * perm_id: Permutation ID (0 for original model, 1+ for permutations)
 - * Additional columns from the permutation test matrix (e.g., R2X, R2Y, Q2, etc.)
- raw_result: The raw ropls opls object from `ropls::opls()`
- meta_data: A list containing metadata from the input experiment

Required packages

This function requires the following packages to be installed:

- ropls for OPLS-DA analysis

See Also

`ropls::opls()`

gly_pca

*Principal Component Analysis (PCA)***Description**

Perform principal component analysis on the expression data. The function uses `prcomp()` to perform PCA and `broom::tidy()` to tidy the results. If `scale = TRUE`, constant variables (zero variance) will be removed before PCA.

Usage

```
gly_pca(
  exp,
  center = TRUE,
  scale = TRUE,
  add_info = TRUE,
  ...,
  log_transform = TRUE
)
```

Arguments

<code>exp</code>	A <code>glyexp::GlycomicsSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information.
<code>center</code>	A logical indicating whether to center the data. Default is <code>TRUE</code> .
<code>scale</code>	A logical indicating whether to scale the data. Default is <code>TRUE</code> .
<code>add_info</code>	A logical value. If <code>TRUE</code> (default), sample and variable information from the experiment will be added to the result tibbles. If <code>FALSE</code> , only the PCA results are returned.
<code>...</code>	Additional arguments passed to <code>prcomp()</code> .
<code>log_transform</code>	A single logical value indicating whether to transform expression values using $\log(x + 1)$ before analysis. Default is <code>TRUE</code> . Set to <code>FALSE</code> for data that are already log-transformed.

Details

By default, the function performs log transformation on the expression data ($\log(x + 1)$) before PCA analysis. Set `log_transform = FALSE` to use the expression values as supplied, for example when they are already log-transformed.

Value

A list containing:

- `tidy_result`: A list of tibbles with PCA results:

- samples: PCA scores for each sample containing the following columns:
 - * sample: Sample name
 - * PC: Principal component name (PC1, PC2, etc.)
 - * value: Score value for the principal component
- variables: PCA loadings for each variable containing the following columns:
 - * variable: Variable name
 - * PC: Principal component name (PC1, PC2, etc.)
 - * value: Loading value for the principal component
- eigenvalues: PCA eigenvalues containing the following columns:
 - * PC: Principal component name (PC1, PC2, etc.)
 - * std.dev: Standard deviation
 - * percent: Percentage of variance explained
 - * cumulative: Cumulative percentage of variance explained
- raw_result: The raw prcomp object from `stats::prcomp()`
- meta_data: A list containing metadata from the input experiment

Required packages

This function only uses base R packages and does not require additional dependencies.

See Also

`stats::prcomp()`

gly_plsda

Partial Least Squares Discriminant Analysis (PLS-DA)

Description

Perform partial least squares discriminant analysis on the expression data. The function uses `ropls::opls()` to perform PLS-DA and returns tidy results.

Usage

```
gly_plsda(  
  exp,  
  group_col = "group",  
  ncomp = 2,  
  scale = TRUE,  
  add_info = TRUE,  
  ...,  
  log_transform = TRUE  
)
```

Arguments

<code>exp</code>	A <code>glyexp::GlycomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information.
<code>group_col</code>	A character string specifying the column name in sample information that contains group labels. Default is "group".
<code>ncomp</code>	An integer indicating the number of components to include. Default is 2.
<code>scale</code>	A logical indicating whether to scale the data. Default is TRUE.
<code>add_info</code>	A logical value. If TRUE (default), sample and variable information from the experiment will be added to the result tibbles. If FALSE, only the PLS-DA results are returned.
<code>...</code>	Additional arguments passed to <code>roppls::opls()</code> .
<code>log_transform</code>	A single logical value indicating whether to transform expression values using $\log(x + 1)$ before analysis. Default is TRUE. Set to FALSE for data that are already log-transformed.

Details

By default, expression values are transformed using $\log(x + 1)$ before analysis. Set `log_transform = FALSE` to use the expression values as supplied, for example when they are already log-transformed.

Value

A list containing:

- `tidy_result`: A list of tibbles with PLS-DA results:
 - `samples`: PLS-DA scores for each sample containing the following columns:
 - * `sample`: Sample name
 - * `group`: Group assignment
 - * `p1`, `p2`, etc.: PLS-DA component scores
 - `variables`: PLS-DA loadings for each variable containing the following columns:
 - * `variable`: Variable name
 - * `p1`, `p2`, etc.: PLS-DA component loadings
 - * `pcorr1`, `pcorr2`, etc.: Correlation between each variable and component
 - `variance`: PLS-DA explained variance containing the following columns:
 - * `component`: Component name (`p1`, `p2`, etc.)
 - * `prop_var_explained`: Proportion of variance explained by each component
 - * `cumulative_prop_var`: Cumulative proportion of variance explained
 - `vip`: Variable Importance in Projection scores containing the following columns:
 - * `variable`: Variable name
 - * `vip`: VIP score
 - `perm_test`: Permutation test results containing the following columns:
 - * `model`: Model type ("Original" for the original model, "Permutation" for permuted models)

- * perm_id: Permutation ID (0 for original model, 1+ for permutations)
- * Additional columns from the permutation test matrix (e.g., R2X, R2Y, Q2, etc.)
- raw_result: The raw ropls opls object from `ropls::opls()`
- meta_data: A list containing metadata from the input experiment

Required packages

This function requires the following packages to be installed:

- ropls for PLS-DA analysis

See Also

[ropls::opls\(\)](#)

gly_roc

ROC Analysis for Glycomics and Glycoproteomics Data

Description

Perform Receiver Operating Characteristic (ROC) analysis for binary classification of glycomics or glycoproteomics data. The function calculates ROC curves and Area Under the Curve (AUC) values for each variable to assess their discriminatory power between two groups.

Usage

```
gly_roc(
  exp,
  group_col = "group",
  pos_class = NULL,
  add_info = TRUE,
  log_transform = FALSE
)
```

Arguments

- | | |
|-----------|---|
| exp | A glyexp::GlycomicSE() or glyexp::GlycoproteomicSE() object, or another SummarizedExperiment containing an expression matrix and sample information. |
| group_col | A character string specifying the column name of the grouping variable in the sample information. Default is "group". The grouping variable must have exactly 2 levels for binary classification. |
| pos_class | A character string specifying which group level should be treated as the positive class. If NULL (default), the second level (alphabetically) will be used as the positive class. |

add_info	A logical value. If TRUE (default), variable information from the experiment will be added to the result tibbles. If FALSE, only the ROC analysis results are returned.
log_transform	A single logical value indicating whether to transform expression values using $\log_2(x + 1e-6)$ before analysis. Default is FALSE. Set to FALSE for data that are already log-transformed.

Details

Expression values are used as supplied by default. Set `log_transform = TRUE` to apply $\log_2(x + 1e-6)$ before analysis.

For each variable, a ROC curve is computed using the expression values as predictor and the binary group labels as response.

The function requires exactly 2 groups in the specified grouping variable. If more than 2 groups are present, an error will be thrown.

Underlying Function:

- ROC analysis is performed using `pROC::roc()`
- Coordinates are extracted using `pROC::coords()`

Value

A list with three elements:

- `tidy_result`: A list containing two tibbles:
 - `auc`: A tibble containing AUC values for each variable with the following columns:
 - * `variable`: Variable name
 - * `auc`: Area Under the Curve value
 - * `auc_ci_low`: Lower bound of 95% confidence interval for AUC
 - * `auc_ci_high`: Upper bound of 95% confidence interval for AUC
 - `coords`: A tibble containing ROC curve coordinates with the following columns:
 - * `variable`: Variable name
 - * `threshold`: Threshold value for classification
 - * `specificity`: Specificity (True Negative Rate)
 - * `sensitivity`: Sensitivity (True Positive Rate)
- `raw_result`: A list of `pROC` objects
- `meta_data`: A list containing metadata from the input experiment The list has classes `glystats_roc_res` and `glystats_res`.

Required packages

This function requires the `pROC` package to be installed for ROC curve computation.

See Also

`pROC::roc()`, `pROC::coords()`

Description

Construct and jointly test correlated variable sets with Hotelling's T^2 statistic.

Usage

```
gly_set_test(
  exp,
  sets = NULL,
  group_col = "group",
  subject_col = NULL,
  method = "hotelling",
  p_adj_method = "BH",
  threshold = 0.9,
  correlation = "pearson",
  clustering = "connected",
  within = NULL,
  log_transform = TRUE
)
```

Arguments

exp	A glyexp::GlycomicSE() or glyexp::GlycoproteomicSE() object, or another SummarizedExperiment containing expression data and sample information.
sets	A uniquely named list of character vectors defining custom sets. When NULL (the default), correlated sets are constructed automatically from exp using threshold, correlation, clustering, and within.
group_col	A character string naming the two-level grouping column in sample information. Default is "group".
subject_col	An optional character string naming subject identifiers. When supplied, one-sample Hotelling tests are applied to paired test-minus-reference difference vectors. Otherwise independent two-sample Hotelling tests are used.
method	Statistical method. Currently only "hotelling" is supported.
p_adj_method	A character string specifying the method used to adjust set-level p-values. See stats::p.adjust() . Default is "BH". If NULL, no adjustment is performed.
threshold	A numeric correlation threshold between 0 and 1 used when sets is NULL. Variables are linked only when their correlation is strictly greater than this value. Default is 0.9.
correlation	Correlation coefficient used when sets is NULL. Either "pearson" (default) or "spearman".

clustering	How correlated variables are grouped when sets is NULL. "connected" (default) allows transitive chaining; "complete" requires every pair in a set to exceed threshold.
within	Optional variable-information columns defining strata within which automatic sets are constructed, such as <code>c("protein", "protein_site")</code> .
log_transform	A single non-missing logical. If TRUE (default), apply $\log_2(x + 1e-6)$ for both automatic set construction and testing. If FALSE, use supplied values without transformation, including values that have already been transformed. Estimates are on the selected scale.

Details

By default, expression values are transformed using $\log_2(x + 1e-6)$. The first factor level of `group_col` is the reference group and the second is the test group. The estimate is therefore test - reference.

Every usable variable is assigned to a set, including a one-variable set when it is not correlated above threshold with another variable. Structurally rank-deficient covariance matrices are standardized to unit member variance and tested in their nonredundant subspace using a Moore-Penrose inverse when the original set dimension satisfies the classical sample-size requirement and the observed mean contrast lies in the estimable subspace. Other ineligible sets are retained with status = "failed", NA statistics, and an explicit `failure_reason`.

Value

A list with classes `glystats_set_test_res` and `glystats_res` containing:

- `tidy_result$sets`: One row per set with estimates for every member, the effective covariance rank in `test_dimension` (NA when rejected before covariance estimation), Hotelling statistic, degrees of freedom, Mahalanobis effect size, p-value, adjusted p-value, and fit status.
- `tidy_result$members`: One row per set member with its marginal estimate and mean within-set correlation.
- `raw_result`: Set definitions, their correlation matrix, individual Hotelling test objects, and automatic set-construction details (when sets is NULL).
- `meta_data`: Metadata copied from `exp`.

See Also

[stats::cor\(\)](#), [stats::hclust\(\)](#), [stats::p.adjust\(\)](#)

Examples

```
set.seed(1)
expression <- matrix(
  stats::rexp(36),
  nrow = 3,
  dimnames = list(c("A", "B", "C"), paste0("S", 1:12))
)
expression["B", ] <- expression["A", ] * exp(stats::rnorm(12, sd = 0.01))
```

```

exp <- SummarizedExperiment::SummarizedExperiment(
  assays = list(expression = expression),
  colData = S4Vectors::DataFrame(
    group = factor(
      rep(c("control", "case"), each = 6),
      levels = c("control", "case")
    )
  )
)
result <- gly_set_test(exp, threshold = 0.8)
result$tidy_result$sets

```

gly_tsne

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Description

Perform t-SNE dimensionality reduction on the expression data. The function uses `Rtsne::Rtsne()` to perform t-SNE analysis.

Usage

```

gly_tsne(
  exp,
  dims = 2,
  perplexity = 30,
  add_info = TRUE,
  ...,
  log_transform = TRUE
)

```

Arguments

exp	A <code>glyexp::GlycomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information.
dims	Number of output dimensions. Default is 2.
perplexity	Perplexity parameter for t-SNE. Default is 30.
add_info	A logical value. If TRUE (default), sample information from the experiment will be added to the result tibble. If FALSE, only the t-SNE coordinates are returned.
...	Additional arguments passed to <code>Rtsne::Rtsne()</code> .
log_transform	A single logical value indicating whether to transform expression values using $\log(x + 1)$ before analysis. Default is TRUE. Set to FALSE for data that are already log-transformed.

Details

By default, expression values are transformed using $\log(x + 1)$ before analysis. Set `log_transform = FALSE` to use the expression values as supplied, for example when they are already log-transformed.

Value

A list with three elements:

- `tidy_result`: A tibble with t-SNE coordinates containing the following columns:
 - `sample`: Sample name
 - `tsne1`: First t-SNE dimension
 - `tsne2`: Second t-SNE dimension
- `raw_result`: The raw Rtsne object
- `meta_data`: A list containing metadata from the input experiment The list has classes `glystats_tsne_res` and `glystats_res`.

Required packages

This function requires the Rtsne package to be installed for t-SNE analysis.

See Also

[Rtsne::Rtsne\(\)](#)

gly_ttest

T-test for Differential Expression Analysis

Description

Perform independent or paired t-tests for glycomics or glycoproteomics data. P-values are adjusted for multiple testing using the method specified by `p_adj_method`.

Usage

```
gly_ttest(
  exp,
  group_col = "group",
  p_adj_method = "BH",
  ref_group = NULL,
  add_info = TRUE,
  ...,
  subject_col = NULL,
  log_transform = TRUE
)
```

Arguments

exp	A <code>glyexp::GlycomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information.
group_col	A character string specifying the column name of the grouping variable in the sample information. Default is "group".
p_adj_method	A character string specifying the method to adjust p-values. See <code>p.adjust.methods</code> for available methods. Default is "BH". If NULL, no adjustment is performed.
ref_group	A character string specifying the reference group. If NULL (default), the first level of the group factor is used as the reference.
add_info	A logical value. If TRUE (default), variable information from the experiment will be added to the result tibble. If FALSE, only the statistical results are returned.
...	Additional arguments passed to <code>stats::t.test()</code> .
subject_col	An optional character string naming the subject identifier column in sample information. When supplied, samples are matched by subject and a paired t-test is performed.
log_transform	A single non-missing logical value. If TRUE (default), transform expression values using $\log_2(x + 1e-6)$ before analysis. If FALSE, use the supplied values unchanged, for example for already log-transformed data.

Details

Tests and effect sizes use the selected analysis scale. Reported fold changes continue to use ratios of group means from the original supplied values.

By default, expression data are transformed using $\log_2(x + 1e-6)$ before statistical testing. Set `log_transform = FALSE` to skip this transformation. Exactly 2 groups are required in the grouping variable. In paired analyses, only subjects observed in both groups are used and `effect_size` is Cohen's d_z based on within-subject differences.

Value

A list with three elements:

- `tidy_result`: A tibble with t-test results containing the following columns:
 - `variable`: Variable name
 - `estimate`: Difference in group means (`group2 - group1`)
 - `estimate1`: Mean of group 1
 - `estimate2`: Mean of group 2
 - `statistic`: t-statistic
 - `p_val`: Raw p-value from t-test
 - `parameter`: Degrees of freedom
 - `conf_low`: Lower bound of 95% confidence interval
 - `conf_high`: Upper bound of 95% confidence interval
 - `effect_size`: Cohen's d , or Cohen's d_z for paired analyses

- method: Statistical method used
- alternative: Alternative hypothesis
- p_adj: Adjusted p-value (if p_adj_method is not NULL)
- log2fc: Log2 fold change ($\log_2(\text{group2_mean} / \text{group1_mean})$)
- raw_result: A list of `t.test` model objects
- meta_data: A list containing metadata from the input experiment The list has classes `glystats_ttest_res` and `glystats_res`.

See Also

`stats::t.test()`

gly_umap

Uniform Manifold Approximation and Projection (UMAP)

Description

Perform UMAP dimensionality reduction on the expression data. The function uses `uwot::umap()` to perform UMAP analysis.

Usage

```
gly_umap(
  exp,
  n_neighbors = 15,
  n_components = 2,
  add_info = TRUE,
  ...,
  log_transform = TRUE
)
```

Arguments

<code>exp</code>	A <code>glyexp::GlycomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information.
<code>n_neighbors</code>	Number of neighbors to consider for each point. Default is 15.
<code>n_components</code>	Number of output dimensions. Default is 2.
<code>add_info</code>	A logical value. If <code>TRUE</code> (default), sample information from the experiment will be added to the result tibble. If <code>FALSE</code> , only the UMAP coordinates are returned.
<code>...</code>	Additional arguments passed to <code>uwot::umap()</code> .
<code>log_transform</code>	A single logical value indicating whether to transform expression values using $\log(x + 1)$ before analysis. Default is <code>TRUE</code> . Set to <code>FALSE</code> for data that are already log-transformed.

Details

By default, expression values are transformed using $\log(x + 1)$ before analysis. Set `log_transform = FALSE` to use the expression values as supplied, for example when they are already log-transformed.

Value

A list with three elements:

- `tidy_result`: A tibble with UMAP coordinates containing the following columns:
 - `sample`: Sample name
 - `umap1`: First UMAP dimension
 - `umap2`: Second UMAP dimension
 - `umap3`, `umap4`, etc.: Additional UMAP dimensions (if `n_components > 2`)
- `raw_result`: The raw UMAP result matrix
- `meta_data`: A list containing metadata from the input experiment The list has classes `glystats_umap_res` and `glystats_res`.

Required packages

This function requires the `uwot` package to be installed for UMAP analysis.

See Also

[uwot::umap\(\)](#)

 gly_wilcox

Wilcoxon Test for Differential Expression Analysis

Description

Perform Wilcoxon rank-sum or signed-rank tests for glycomics or glycoproteomics data. P-values are adjusted for multiple testing using the method specified by `p_adj_method`.

Usage

```
gly_wilcox(
  exp,
  group_col = "group",
  p_adj_method = "BH",
  ref_group = NULL,
  add_info = TRUE,
  ...,
  subject_col = NULL,
  log_transform = TRUE
)
```

Arguments

<code>exp</code>	A <code>glyexp::GlycomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object, or another <code>SummarizedExperiment</code> containing an expression matrix and sample information.
<code>group_col</code>	A character string specifying the column name of the grouping variable in the sample information. Default is "group".
<code>p_adj_method</code>	A character string specifying the method to adjust p-values. See <code>p.adjust.methods</code> for available methods. Default is "BH". If NULL, no adjustment is performed.
<code>ref_group</code>	A character string specifying the reference group. If NULL (default), the first level of the group factor is used as the reference.
<code>add_info</code>	A logical value. If TRUE (default), variable information from the experiment will be added to the result tibble. If FALSE, only the statistical results are returned.
<code>...</code>	Additional arguments passed to <code>stats::wilcox.test()</code> .
<code>subject_col</code>	An optional character string naming the subject identifier column in sample information. When supplied, samples are matched by subject and a paired Wilcoxon signed-rank test is performed.
<code>log_transform</code>	A single non-missing logical value. If TRUE (default), transform expression values using $\log_2(x + 1e-6)$ before analysis. If FALSE, use the supplied values unchanged, for example for already log-transformed data.

Details

Tests and effect sizes use the selected analysis scale. Reported fold changes continue to use ratios of group means from the original supplied values.

By default, expression data are transformed using $\log_2(x + 1e-6)$ before statistical testing. Set `log_transform = FALSE` to skip this transformation. Exactly 2 groups are required in the grouping variable. In paired analyses, only subjects observed in both groups are used and `effect_size` is the matched-pairs rank-biserial correlation.

Value

A list with three elements:

- `tidy_result`: A tibble with Wilcoxon test results containing the following columns:
 - `variable`: Variable name
 - `statistic`: Wilcoxon test statistic
 - `p_val`: Raw p-value from Wilcoxon test
 - `effect_size`: Rank-biserial correlation, using matched pairs in paired analyses
 - `method`: Statistical method used
 - `alternative`: Alternative hypothesis
 - `p_adj`: Adjusted p-value (if `p_adj_method` is not NULL)
 - `log2fc`: Log2 fold change ($\log_2(\text{group2_mean} / \text{group1_mean})$) Additional columns from experiment metadata may be included if `add_info = TRUE`.
- `raw_result`: A list of `wilcox.test` model objects

- `meta_data`: A list containing metadata from the input experiment The list has classes `glystats_wilcox_res` and `glystats_res`.

See Also

`stats::wilcox.test()`

Index

emmeans::contrast(), 8
emmeans::emmeans(), 8

filter_sig_vars, 2
FSA::dunnTest(), 19

get_raw_result(get_tidy_result), 5
get_tidy_result, 5
gly_ancova, 6
gly_ancova(), 2
gly_anova, 8
gly_anova(), 2, 5, 6
gly_cor, 10
gly_cox, 12
gly_fold_change, 13
gly_hclust, 14
gly_kmeans, 16
gly_kruskal, 18
gly_kruskal(), 2, 5
gly_limma, 20
gly_limma(), 2, 5, 23
gly_linear_model, 22
gly_linear_model(), 2, 5
gly_oplsda, 24
gly_pca, 26
gly_plsda, 27
gly_roc, 29
gly_set_test, 31
gly_tsne, 33
gly_ttest, 34
gly_ttest(), 2, 6
gly_umap, 36
gly_wilcox, 37
gly_wilcox(), 2
glyexp::GlycomicSE(), 4, 7, 9, 11, 12,
14–16, 18, 20, 22, 24, 26, 28, 29, 31,
33, 35, 36, 38
glyexp::GlycoproteomicSE(), 4, 7, 9, 11,
12, 14–16, 18, 20, 22, 24, 26, 28, 29,
31, 33, 35, 36, 38

Hmisc::rcorr(), 12

limma::eBayes(), 22, 23
limma::lmFit(), 22, 23
limma::makeContrasts(), 22

pROC::coords(), 30
pROC::roc(), 30

ropls::opls(), 25, 29
Rtsne::Rtsne(), 34

stats::aov(), 8, 10
stats::cor(), 12, 32
stats::cutree(), 16
stats::dist(), 16
stats::friedman.test(), 19
stats::hclust(), 16, 32
stats::kmeans(), 17
stats::kruskal.test(), 19
stats::model.matrix(), 23
stats::p.adjust(), 22, 31, 32
stats::prcomp(), 27
stats::t.test(), 36
stats::TukeyHSD(), 10
stats::wilcox.test(), 39
survival::coxph(), 13
survival::Surv(), 13

uwot::umap(), 37