Package: glystats (via r-universe)

May 22, 2026

Title Statistical Analysis of Glycoproteomics and Glycomics Data

Version 0.10.1

Description Provides a unified toolbox for bioinformatics analyses of glycomics and glycoproteomics data. Implemented methods include differential testing (t-test, Wilcoxon rank-sum test, ANOVA, Kruskal–Wallis), multivariate modeling and visualization (PCA, t-SNE, UMAP), supervised learning (PLS-DA, OPLS-DA), clustering (k-means, hierarchical, consensus clustering), network analysis (weighted gene co-expression network analysis, WGCNA), survival modeling (Cox proportional hazards), enrichment analysis, correlation analysis, and ROC/AUC evaluation. All user-facing functions follow the gly_*() naming convention to facilitate auto-completion in RStudio. The package is designed to work seamlessly with 'glyexp' and tidy data workflows.

License MIT + file LICENSE

Suggests testthat (>= 3.0.0), FSA, emmeans, ropls, mockery, Rtsne, org.Hs.eg.db, uwot, pROC, clusterProfiler, ReactomePA, DOSE, ggdendro, Hmisc, limma, WGCNA, ConsensusClusterPlus, 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 (>= 0.12.3), rlang, lifecycle, tidyr, dplyr, tidyselect, magrittr, janitor, checkmate, cli, stats, broom, purrr, stringr, withr, survival, glyrepr (>= 0.9.0)

Depends R (>= 4.1)

VignetteBuilder knitr

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

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

Date/Publication 2026-05-20 02:31:07 UTC

RemoteUrl https://github.com/glycoverse/glystats

RemoteRef v0.10.1

RemoteSha a2bbb687a5d870c5609e4763ff5ffca9bd133ff8

Contents

`filter_sig_vars` *Filter Significant Variables*

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()`, and `gly_limma()`.

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,
  ...
```

```
)

## 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,
  ...
)
```

Arguments

`exp`	An `glyexp::experiment()`. Please use the same experiment 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 `gly_anova()` and `gly_kruskal()` 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 `gly_anova()`, `gly_kruskal()`, and `gly_limma()` results only) Specifies which comparison to filter on. A string with the format "group1_vs_group2". For `gly_anova()` and `gly_kruskal()` results, `comparison` is only used when `on` is "post_hoc_test". If not provided, filtering will be performed on the main test results for `gly_anova()` and `gly_kruskal()`, and variables will be kept if they are significant in any comparison for `gly_limma()`.

Value

An new glyexp::experiment() object.

Examples

```
library(glyexp)
library(glyclean)

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

---

get_tidy_result *Get the result of a glystats analysis*

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

`res`	A glystats result object.
`which`	Used to specify which element to get, when the result is a list. For glystats results with only one tidy result (e.g. gly_ttest()), this argument is not needed. For others (e.g. gly_anova()), 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_var(n = 10)
```

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

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

`gly_ancova` *Analysis of Covariance (ANCOVA) for Differential Expression Analysis*

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,
  ...
)

gly_ancova_(expr_mat, groups, covariates, p_adj_method = "BH", ...)
```

Arguments

`exp`	A `glyexp::experiment()` object containing expression matrix and sample information.
`group_col`	(Only for `gly_ancova()`) A character string specifying the column name of the grouping variable in the sample information. Default is `"group"`.
`covariate_cols`	(Only for `gly_ancova()`) 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 p.adjust.methods 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. Only applicable to gly_ancova().
...	Additional arguments passed to stats::aov().
expr_mat	(Only for gly_ancova_()) A numeric matrix with variables as rows and samples as columns.
groups	(Only for gly_ancova_()) A factor or character vector specifying group membership for each sample. Must have at least 2 levels. Character vectors will be automatically converted to factors.
covariates	(Only for gly_ancova_()) A data frame, matrix, or vector of sample-level covariates. At least one covariate must be provided.

Details

The function performs log2 transformation on the expression data (log2(x + 1e-6)) before statistical testing. At least 2 groups and at least 1 covariate are required.

gly_ancova() is the top-level API that works with glyexp::experiment() objects and supports the add_info parameter for joining experiment metadata.

gly_ancova_() is the underlying API that works with matrices, factor vectors, and covariate data directly, providing more flexibility for users who don't use the glyexp package.

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 ANOVA for Differential Expression Analysis*

Description

Perform one-way ANOVA for glycomics or glycoproteomics data. The function supports parametric comparison of multiple groups. 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, ...)

gly_anova_(expr_mat, groups, p_adj_method = "BH", ...)
```

Arguments

`exp`	A `glyexp::experiment()` object containing expression matrix and sample information.
`group_col`	(Only for `gly_anova()`) 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 `p.adjust.methods` 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. Only applicable to `gly_anova()`.

`...`	Additional arguments passed to `stats::aov()`.
`expr_mat`	(Only for `gly_anova_()`) A numeric matrix with variables as rows and samples as columns.
`groups`	(Only for `gly_anova_()`) A factor or character vector specifying group membership for each sample. Must have at least 2 levels. Character vectors will be automatically converted to factors.

Details

The function performs log2 transformation on the expression data (log2(x + 1e-6)) before statistical testing. At least 2 groups are required in the grouping variable.

`gly_anova()` is the top-level API that works with `glyexp::experiment()` objects and supports the `add_info` parameter for joining experiment metadata.

`gly_anova_()` is the underlying API that works with matrices and factor vectors directly, providing more flexibility for users who don't use the glyexp package.

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).

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
    - `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 p-value from Tukey's HSD test
    - `p_adj`: Adjusted p-value from Tukey's HSD test
- `raw_result`: A list containing:

  - `main_test`: A list of raw `aov` model objects.
  - `post_hoc_test`: A list of raw `TukeyHSD` objects. Post-hoc comparison labels follow the package direction, i.e. `test_group - ref_group`.
- `meta_data` (only for `gly_anova()`): 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", ...)

gly_cor_(
  expr_mat,
  on = "variable",
  method = "pearson",
  p_adj_method = "BH",
  ...
)
```

Arguments

`exp`	A `glyexp::experiment()` object containing 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 Hmisc::rcorr.
`p_adj_method`	A character string specifying the method to adjust p-values. See `p.adjust.methods` for available methods. Default is "BH". If NULL, no adjustment is performed.
`...`	Additional arguments passed to `Hmisc::rcorr()`.
`expr_mat`	(Only for `gly_cor_()`) A numeric matrix with variables as rows and samples as columns.

Details

The function performs log2 transformation on the expression data (log2(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.

`gly_cor()` is the top-level API that works with `glyexp::experiment()` objects

`gly_cor_()` is the underlying API that works with matrices directly, providing more flexibility for users who don't use the glyexp package.

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` (only for `gly_cor()`): 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,
  ...
)

gly_cox_(expr_mat, time, event, p_adj_method = "BH", ...)
```

Arguments

exp	A `glyexp::experiment()` object containing 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 `p.adjust.methods` 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. Only applicable to `gly_cox()`.
...	Additional arguments passed to `survival::coxph()`.
expr_mat	(Only for `gly_cox_()`) A numeric matrix with variables as rows and samples as columns.
time	A numeric vector specifying the survival time for each sample.
event	A numeric vector specifying the event indicator for each sample (1 for event, 0 for censoring).

Details

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(coefficient))
  - `p_adj`: Adjusted p-value (if p_adj_method is not NULL)
- `raw_result`: A list of raw `coxph` model objects.
- `meta_data` (only for `gly_cox()`): A list containing metadata from the input experiment

See Also

`survival::coxph()`, `survival::Surv()`

`gly_fold_change` *Calculate fold change*

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)

gly_fold_change_(expr_mat, groups)
```

Arguments

`exp`	A `glyexp::experiment()` object.
`group_col`	(Only for `gly_fold_change()`) The column name of the group information in the sample information.
`add_info`	A logical value. If TRUE (default), variable information from the experiment will be added to the result tibble. If FALSE, only the fold change results are returned. Only applicable to `gly_fold_change()`.
`expr_mat`	(Only for `gly_fold_change_()`) A numeric matrix with variables as rows and samples as columns.
`groups`	(Only for `gly_fold_change_()`) A factor or character vector specifying group membership for each sample. Character vectors will be automatically converted to factors. If two groups, the first level is the reference group. If more than two groups, pairwise comparisons will be performed, with levels coming first as reference groups.

Details

`gly_fold_change()` is the top-level API that works with `glyexp::experiment()` objects and supports the `add_info` parameter for joining experiment metadata.

`gly_fold_change_()` is the underlying API that works with matrices and factor vectors directly, providing more flexibility for users who don't use the glyexp package.

Value

A list with three elements:

- `tidy_result`: A tibble with fold change results containing the following columns:
  - `variable`: Variable name
  - `log2fc`: Log2 fold change (log2(group2_mean / group1_mean)) 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` (only for `gly_fold_change()`): 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,
  ...
)

gly_hclust_(
  expr_mat,
  on = "variable",
  k_values = c(2, 3, 4, 5),
  scale = TRUE,
  ...
)
```

Arguments

exp	A `glyexp::experiment()` object containing 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 c(2, 3, 4, 5). If NULL, no cluster assignments are returned.
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. Only applicable to `gly_hclust()`.
...	Additional arguments passed to `stats::dist()` and `stats::hclust()`. Note: if both functions need a `method` parameter, use `dist.method` for distance and `hclust.method` for clustering method.
expr_mat	(Only for `gly_hclust_()`) A numeric matrix with variables as rows and samples as columns.

Details

The function performs log2 transformation on the expression data (log2(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.

`gly_hclust()` is the top-level API that works with `glyexp::experiment()` objects and supports the `add_info` parameter for joining experiment metadata.

`gly_hclust_()` is the underlying API that works with matrices directly, providing more flexibility for users who don't use the glyexp package.

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 ggdendro is available) containing:
    * `x`, `y`: Position coordinates
    * `label`: Label text
- `raw_result`: The raw hclust object from `stats::hclust()`
- `meta_data` (only for `gly_hclust()`): 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,
  ...
)

gly_kmeans_(expr_mat, on = "variable", centers = 3, scale = TRUE, ...)
```

Arguments

`exp`	A `glyexp::experiment()` object containing 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. Only applicable to `gly_kmeans()`.
`...`	Additional arguments passed to `stats::kmeans()`.
`expr_mat`	(Only for `gly_kmeans_()`) A numeric matrix with variables as rows and samples as columns.

Details

The function performs log2 transformation on the expression data (log2(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.

`gly_kmeans()` is the top-level API that works with `glyexp::experiment()` objects and supports the `add_info` parameter for joining experiment metadata.

`gly_kmeans_()` is the underlying API that works with matrices directly, providing more flexibility for users who don't use the glyexp package.

Data Preparation: Data is log2-transformed 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` (only for `gly_kmeans()`): 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 test for Differential Expression Analysis*

Description

Perform Kruskal-Wallis test for glycomics or glycoproteomics data. The function supports non-parametric comparison of multiple groups. 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,
  ...
)

gly_kruskal_(expr_mat, groups, p_adj_method = "BH", ...)
```

Arguments

`exp`	A `glyexp::experiment()` object containing expression matrix and sample information.
`group_col`	(Only for `gly_kruskal()`) 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 `p.adjust.methods` 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. Only applicable to `gly_kruskal()`.
`...`	Additional arguments passed to `stats::kruskal.test()`.
`expr_mat`	(Only for `gly_kruskal_()`) A numeric matrix with variables as rows and samples as columns.
`groups`	(Only for `gly_kruskal_()`) A factor or character vector specifying group membership for each sample. Must have at least 2 levels. Character vectors will be automatically converted to factors.

Details

The function performs log2 transformation on the expression data (log2(x + 1e-6)) before statistical testing. At least 2 groups are required in the grouping variable.

`gly_kruskal()` is the top-level API that works with `glyexp::experiment()` objects and supports the `add_info` parameter for joining experiment metadata.

`gly_kruskal_()` is the underlying API that works with matrices and factor vectors directly, providing more flexibility for users who don't use the glyexp package.

For any variable failed to fit a `stats::kruskal.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_adj < 0.05).

Value

A list containing three elements:

- `tidy_result`: A list containing:
  - `main_test`: A tibble with Kruskal-Wallis 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
    - `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 p-value from Dunn's test
    - `p_adj`: Adjusted p-value from Dunn's test
- `raw_result`: A list containing:
  - `main_test`: A list of raw `kruskal.test` objects.
  - `post_hoc_test`: A list of raw `dunnTest` objects. Post-hoc comparison labels and direction-sensitive statistics follow the package direction, i.e. `test_group - ref_group`.
- `meta_data`: A list containing metadata from the input experiment.

Required packages

This function requires the `FSA` package for Dunn's post-hoc test.

See Also

stats::kruskal.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,
  ...
)

gly_limma_(
  expr_mat,
  groups,
  p_adj_method = "BH",
  ref_group = NULL,
  contrasts = NULL,
  covariates = NULL,
  subjects = NULL,
  ...
)
```

Arguments

exp	A glyexp_experiment object containing expression data and sample information.
group_col	(Only for gly_limma()) A character string specifying the column name in sample information that contains group labels. Default is "group".
covariate_cols	(Only for gly_limma()) A character vector specifying column names in sample information to include as covariates in the limma model. Default is NULL.

`subject_col` (Only for `gly_limma()`) 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 `stats::p.adjust()`. 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.

`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. Only applicable to `gly_limma()`.

`...` Additional arguments passed to `limma::lmFit()`.

`expr_mat` (Only for `gly_limma_()`) A numeric matrix with variables as rows and samples as columns.

`groups` (Only for `gly_limma_()`) A factor or character vector specifying group membership for each sample. Must have at least 2 levels. Character vectors will be automatically converted to factors. If `contrasts` is not provided, the levels coming first in the factor will be used as the reference group.

`covariates` (Only for `gly_limma_()`) A data frame, matrix, or vector of sample-level covariates. Must have the same number of rows as `expr_mat` has columns. If row names are provided and match `colnames(expr_mat)`, they will be aligned automatically. Default is NULL.

`subjects` (Only for `gly_limma_()`) A vector or factor of subject identifiers for paired comparisons. Must have length equal to `ncol(expr_mat)`. If names or row names are provided and match `colnames(expr_mat)`, they will be aligned automatically. Default is NULL.

Details

The function performs log2 transformation on the expression data (log2(x + 1e-6)) before statistical testing. The analysis uses linear models with empirical Bayes moderation to improve statistical power, especially for small sample sizes.

`gly_limma()` is the top-level API that works with `glyexp::experiment()` objects and supports the `add_info` parameter for joining experiment metadata.

`gly_limma_()` is the underlying API that works with matrices and factor vectors directly, providing more flexibility for users who don't use the glyexp package.

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 `~ 0 + group + 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`: Log2 fold change
  - `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` (only for `gly_limma()`): 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_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,
  ...
)

gly_oplsda_(expr_mat, groups, pred_i = 1, ortho_i = NA, scale = TRUE, ...)
```

Arguments

exp	A `glyexp::experiment()` object containing expression matrix and sample information.
group_col	(Only for `gly_oplsda()`) A character string specifying the column name in sample information that contains group labels. Default is "group".
pred_i	An integer indicating the number of predictive components to include. Default is 1.
ortho_i	An integer indicating the number of orthogonal components to include. Default is NA (automatic).
scale	A logical indicating whether to scale the data. Default is TRUE.
add_info	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. Only applicable to `gly_oplsda()`.
...	Additional arguments passed to `ropls::opls()`.
expr_mat	(Only for `gly_oplsda_()`) A numeric matrix with variables as rows and samples as columns.
groups	(Only for `gly_oplsda_()`) A factor or character vector specifying group membership for each sample. Must have exactly 2 levels. Character vectors will be automatically converted to factors.

Details

`gly_oplsda()` is the top-level API that works with `glyexp::experiment()` objects and supports the `add_info` parameter for joining experiment metadata.

`gly_oplsda_()` is the underlying API that works with matrices and factor vectors directly, providing more flexibility for users who don't use the glyexp package.

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` (only for `gly_oplsda()`): 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, ...)

gly_pca_(expr_mat, center = TRUE, scale = TRUE, ...)
```

Arguments

`exp`	A `glyexp::experiment()` object containing expression matrix and sample information.
`center`	A logical indicating whether to center the data. Default is TRUE.
`scale`	A logical indicating whether to scale the data. Default is TRUE.
`add_info`	A logical value. If TRUE (default), sample and variable information from the experiment will be added to the result tibbles. If FALSE, only the PCA results are returned. Only applicable to `gly_pca()`.
`...`	Additional arguments passed to `prcomp()`.
`expr_mat`	(Only for `gly_pca_()`) A numeric matrix with variables as rows and samples as columns.

Details

The function performs log transformation on the expression data (log(x + 1)) before PCA analysis.

`gly_pca()` is the top-level API that works with `glyexp::experiment()` objects and supports the `add_info` parameter for joining experiment metadata.

`gly_pca_()` is the underlying API that works with matrices directly, providing more flexibility for users who don't use the glyexp package.

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` (only for `gly_pca()`): 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,
  ...
)

gly_plsda_(expr_mat, groups, ncomp = 2, scale = TRUE, ...)
```

Arguments

exp	A glyexp::experiment() object containing expression matrix and sample information.
group_col	(Only for gly_plsda()) A character string specifying the column name in sample information that contains group labels. Default is "group".
ncomp	An integer indicating the number of components to include. Default is 2.
scale	A logical indicating whether to scale the data. Default is TRUE.
add_info	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. Only applicable to gly_plsda().
...	Additional arguments passed to ropls::opls().
expr_mat	(Only for gly_plsda_()) A numeric matrix with variables as rows and samples as columns.
groups	(Only for gly_plsda_()) A factor or character vector specifying group membership for each sample. Character vectors will be automatically converted to factors.

Details

gly_plsda() is the top-level API that works with glyexp::experiment() objects and supports the add_info parameter for joining experiment metadata.

gly_plsda_() is the underlying API that works with matrices and factor vectors directly, providing more flexibility for users who don't use the glyexp package.

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 (only for gly_plsda()): 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)

gly_roc_(expr_mat, groups, pos_class = NULL)
```

Arguments

exp	A glyexp::experiment() object containing expression matrix and sample information.
group_col	(Only for gly_roc()) 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. Only applicable to gly_roc().
expr_mat	(Only for gly_roc_()) A numeric matrix with variables as rows and samples as columns.
groups	(Only for gly_roc_()) A factor or character vector specifying group membership for each sample. Must have exactly 2 levels. Character vectors will be automatically converted to factors.

Details

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.

gly_roc() is the top-level API that works with glyexp::experiment() objects and supports the add_info parameter for joining experiment metadata.

gly_roc_() is the underlying API that works with matrices and factor vectors directly, providing more flexibility for users who don't use the glyexp package.

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 (only for gly_roc()): 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()

---

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, ...)

gly_tsne_(expr_mat, dims = 2, perplexity = 30, ...)
```

Arguments

`exp`	A `glyexp::experiment()` object containing 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. Only applicable to `gly_tsne()`.
`...`	Additional arguments passed to `Rtsne::Rtsne()`.
`expr_mat`	(Only for gly_tsne_()) A numeric matrix with variables as rows and samples as columns.

Details

`gly_tsne()` is the top-level API that works with `glyexp::experiment()` objects and supports the `add_info` parameter for joining experiment metadata.

`gly_tsne_()` is the underlying API that works with matrices directly, providing more flexibility for users who don't use the glyexp package.

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` (only for gly_tsne()): 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` *Two-sample t-test for Differential Expression Analysis*

Description

Perform two-sample t-test for glycomics or glycoproteomics data. The function supports Student's t-test for comparing two groups. 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,
  ...
)

gly_ttest_(expr_mat, groups, p_adj_method = "BH", ref_group = NULL, ...)
```

Arguments

`exp`	A `glyexp::experiment()` object containing expression matrix and sample information.
`group_col`	(Only for `gly_ttest()`) 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 `p.adjust.methods` 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. Only applicable to `gly_ttest()`.
`...`	Additional arguments passed to `stats::t.test()`.
`expr_mat`	(Only for `gly_ttest_()`) A numeric matrix with variables as rows and samples as columns.

`groups` (Only for `gly_ttest_()`) A factor or character vector specifying group membership for each sample. Must have exactly 2 levels. Character vectors will be automatically converted to factors.

Details

The function performs log2 transformation on the expression data (log2(x + 1e-6)) before statistical testing. Exactly 2 groups are required in the grouping variable.

`gly_ttest()` is the top-level API that works with `glyexp::experiment()` objects and supports the `add_info` parameter for joining experiment metadata.

`gly_ttest_()` is the underlying API that works with matrices and factor vectors directly, providing more flexibility for users who don't use the glyexp package.

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
  - `method`: Statistical method used
  - `alternative`: Alternative hypothesis
  - `p_adj`: Adjusted p-value (if p_adj_method is not NULL)
  - `log2fc`: Log2 fold change (log2(group2_mean / group1_mean))
- `raw_result`: A list of `t.test` model objects
- `meta_data` (only for `gly_ttest()`): 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, ...)

gly_umap_(expr_mat, n_neighbors = 15, n_components = 2, ...)
```

Arguments

`exp`	A `glyexp::experiment()` object containing expression matrix and sample information.
`n_neighbors`	Number of neighbors to consider for each point. Default is 15.
`n_components`	Number of output dimensions. Default is 2.
`add_info`	A logical value. If TRUE (default), sample information from the experiment will be added to the result tibble. If FALSE, only the UMAP coordinates are returned. Only applicable to `gly_umap()`.
`...`	Additional arguments passed to `uwot::umap()`.
`expr_mat`	(Only for `gly_umap_()`) A numeric matrix with variables as rows and samples as columns.

Details

`gly_umap()` is the top-level API that works with `glyexp::experiment()` objects and supports the `add_info` parameter for joining experiment metadata.

`gly_umap_()` is the underlying API that works with matrices directly, providing more flexibility for users who don't use the glyexp package.

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` (only for `gly_umap()`): 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 rank-sum test for Differential Expression Analysis*

Description

Perform Wilcoxon rank-sum test (Mann-Whitney U test) for glycomics or glycoproteomics data. The function supports non-parametric comparison of two groups. 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,
  ...
)

gly_wilcox_(expr_mat, groups, p_adj_method = "BH", ref_group = NULL, ...)
```

Arguments

`exp`	A `glyexp::experiment()` object containing expression matrix and sample information.
`group_col`	(Only for `gly_wilcox()`) 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 `p.adjust.methods` 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. Only applicable to `gly_wilcox()`.
`...`	Additional arguments passed to `stats::wilcox.test()`.
`expr_mat`	(Only for `gly_wilcox_()`) A numeric matrix with variables as rows and samples as columns.

`groups` (Only for `gly_wilcox_()`) A factor or character vector specifying group membership for each sample. Must have exactly 2 levels. Character vectors will be automatically converted to factors.

Details

The function performs log2 transformation on the expression data (log2(x + 1e-6)) before statistical testing. Exactly 2 groups are required in the grouping variable.

`gly_wilcox()` is the top-level API that works with `glyexp::experiment()` objects and supports the `add_info` parameter for joining experiment metadata.

`gly_wilcox_()` is the underlying API that works with matrices and factor vectors directly, providing more flexibility for users who don't use the glyexp package.

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
  - `method`: Statistical method used
  - `alternative`: Alternative hypothesis
  - `p_adj`: Adjusted p-value (if p_adj_method is not NULL)
  - `log2fc`: Log2 fold change (log2(group2_mean / 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` (only for `gly_wilcox()`): A list containing metadata from the input experiment The list has classes `glystats_wilcox_res` and `glystats_res`.

See Also

`stats::wilcox.test()`

Index