Package: glysmith (via r-universe)

June 7, 2026

Title Full-Featured Analysis Pipeline for Glycomics and Glycoproteomics

Version 0.11.0

Description glysmith provides high-level, end-to-end workflows for glycomics and glycoproteomics data analysis within the glycoverse ecosystem. It acts as an orchestrator that integrates data cleaning, quality control, derived trait computation, motif detection, statistical testing, and visualization into unified, one-command analytical pipelines. Featuring AI-assisted pipeline generation, glysmith can intelligently translate natural language research questions into executable analysis blueprints, streamlining complex bioinformatics workflows. Built on top of the experiment() data container and domain-knowledge-aware infrastructure provided by glyclean, glydet, glymotif, glystats, glyvis, and related packages, glysmith enables users to quickly forge polished tables, figures, and analysis reports suitable for publication. The package is designed for reproducibility and ease of use, allowing both novice and advanced users to obtain standardized and structure-aware results with minimal code while retaining full flexibility for customization.

License MIT + file LICENSE

Suggests glyclean (>= 0.10.1), glyanno (>= 0.4.2), glydb (>= 0.4.0), glydet (>= 0.10.1), glyfun (>= 0.1.0), glyrepr (>= 0.11.0), glystats (>= 0.6.2), glyvis (>= 0.5.0), glymotif (>= 0.13.0), knitr, withr, pROC, Rtsne, uwot, EnhancedVolcano, enrichplot, org.Hs.eg.db, clusterProfiler, ReactomePA, ggplotify, pheatmap, factoextra, FSA, ggseqlogo, missForest, Hmisc, GGally, survival, UniProt.ws, ropls, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://glycoverse.github.io/glysmith/

Imports checkmate, cli, ellmer, dplyr, fs, ggplot2, glyexp (>= 0.11.0), glue, jsonlite, purrr, readr, rlang, rmarkdown, stringr, tibble

Depends R (>= 4.1.0)

VignetteBuilder knitr

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libuv1-dev libxml2-dev libssl-dev libx11-dev

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

Date/Publication 2026-05-08 03:21:48 UTC

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

RemoteRef v0.11.0

RemoteSha 0f0f8d230d9ed33ba484ab37fec920d48dacab27

Contents

`blueprint` *Create a Blueprint*

Description

A blueprint is a list of steps that are executed in order. Type `step_` and TAB in RStudio to see all available steps.

Usage

```
blueprint(...)
```

Arguments

`...`	One or more step objects.

Value

A blueprint object.

Examples

```
blueprint(
  step_preprocess(),
  step_pca(),
  step_dea_limma(),  # this comma is ok
)
```

`blueprint_default` *Default blueprint*

Description

This blueprint contains the following steps:

- step_ident_overview(): Summarize the experiment using `glyexp::summarize_experiment()`.
- step_preprocess(): Preprocess the data using `glyclean::auto_clean()`.
- step_plot_qc(when = "post"): Plot QC plots using `glyclean::plot_qc()`.
- step_pca(): Principal component analysis using `glystats::gly_pca()`, and plot the PCA using `glyvis::plot_pca()`.
- step_dea_limma(): Differential analysis using `glystats::gly_limma()`.
- step_volcano(): Plot a volcano plot using `glyvis::plot_volcano()`.
- step_heatmap(on = "sig_exp"): Plot a heatmap using `glyvis::plot_heatmap()`.
- step_sig_enrich_go(): Perform GO enrichment analysis using `glyfun::enrich_ora_go()`.
- step_sig_enrich_kegg(): Perform KEGG enrichment analysis using `glyfun::enrich_ora_kegg()`.
- step_sig_enrich_reactome(): Perform Reactome enrichment analysis using `glyfun::enrich_ora_reactome()`.
- step_derive_traits(): Derive traits using `glydet::derive_traits()`.
- step_dea_limma(on = "trait_exp"): Differential trait analysis using `glystats::gly_limma()`.
- step_heatmap(on = "sig_trait_exp"): Plot a heatmap using `glyvis::plot_heatmap()`.

Usage

```
blueprint_default(preprocess = TRUE, enrich = TRUE, traits = TRUE)
```

Arguments

`preprocess`	Whether to include `step_preprocess()`.
`enrich`	Whether to include the enrichment steps, i.e. `step_sig_enrich_go()`, `step_sig_enrich_kegg()`, and `step_sig_enrich_reactome()`.
`traits`	Whether to include the derived trait analysis steps, i.e. `step_derive_traits()` and `step_dea_limma(on = "trait_exp")`.

Value

A `glysmith_blueprint` object.

Examples

```
blueprint_default()
```

br *Create a Branch in a Blueprint*

Description

Use `br()` to group steps that should run as an isolated branch with namespaced outputs prefixed by `<name>__`.

Usage

```
br(name, ...)
```

Arguments

`name`	Branch name used as a prefix for outputs.
`...`	One or more step objects.

Value

A branch object used inside `blueprint()`.

Examples

```
blueprint(
  step_preprocess(),
  br("limma",
    step_dea_limma(),
    step_volcano()
  ),
  br("ttest",
    step_dea_ttest(),
    step_volcano()
  )
)
```

cast_exp *Get Data from GlySmith Result*

Description

Helper functions to get processed experiment, plots, tables or data from a glysmith result object.

Usage

```
cast_exp(x)

cast_plot(x, name = NULL)

cast_table(x, name = NULL)

cast_data(x, name = NULL)
```

Arguments

x	A glysmith result object.
name	The name of the plot or table to get. If not specified, return available names.

Value

- `cast_exp()`: a `glyexp::experiment()`.
- `cast_plot()`: a `ggplot2::ggplot()`.
- `cast_table()`: a `tibble::tibble()`.
- `cast_data()`: can be any R object.

Examples

```
## Not run:
library(glyexp)
exp <- real_experiment2
result <- forge_analysis(exp)
cast_exp(result)
cast_table(result)
cast_table(result, "summary")

## End(Not run)
```

---

check_glysmith_deps *Check glysmith dependencies for a blueprint*

Description

Checks whether the packages required by steps in a blueprint are installed. This does not install or check every package listed in `Suggests`; it only checks the packages declared by the steps in `blueprint`.

Usage

```
check_glysmith_deps(
  blueprint = blueprint_default(),
  action = c("ask", "error", "note")
)
```

Arguments

blueprint	A blueprint(). Defaults to blueprint_default().
action	Character string indicating what to do if packages are missing: • "ask" (default): Prompt the user to install missing packages • "error": Throw an error if any packages are missing • "note": Just print which packages are missing, don't prompt

Value

Returns TRUE invisibly if all packages are installed. If action = "ask", may return TRUE after installation or FALSE if user declines.

Examples

```
## Not run:
# Check dependencies required by the default blueprint
check_glysmith_deps()

# Check dependencies required by a custom blueprint
bp <- blueprint(
  step_ident_overview(),
  step_pca()
)
check_glysmith_deps(bp)

## End(Not run)
```

---

forge_analysis *Perform the Whole Analysis Pipeline*

Description

This function performs a comprehensive analysis for group comparison.

Usage

```
forge_analysis(exp, blueprint = blueprint_default(), group_col = "group")
```

Arguments

exp	A glyexp::experiment() object.
blueprint	A glysmith_blueprint object. Default is blueprint_default().
group_col	Column name of group information in the sample information. Used for various analyses. Default is "group".

Value

A glysmith_result object, with the following components:

- exp: the experiment after preprocessing.
- plots: a named list of ggplot objects.
- tables: a named list of tibbles.
- meta: a named list of metadata, containing:
  - explanation: a named character vector or list of explanations for each plot and table, with keys like tables$summary and plots$pca.
  - steps: a character vector of the steps of the analysis.
  - log: the messages and outputs from each step.
- blueprint: the blueprint used for the analysis.

Examples

```
## Not run:
exp <- glyexp::real_experiment2
result <- forge_analysis(exp)
print(result)

## End(Not run)
```

---

inquire_blueprint *Create a Blueprint using Natural Language*

Description

[Experimental] Ask a Large Language Model (LLM) to create a blueprint for glycomics or glycoproteomics data analysis. DeepSeek is used by default for backward compatibility. Other ellmer providers can be selected with provider, model, and provider-specific API key configuration.

Usage

```
inquire_blueprint(
  description,
  exp = NULL,
  group_col = "group",
  model = getOption("glysmith.ai_model", NULL),
  max_retries = 3,
  provider = getOption("glysmith.ai_provider", "deepseek"),
  api_key = getOption("glysmith.ai_api_key", NULL),
  base_url = getOption("glysmith.ai_base_url", NULL)
)
```

Arguments

description	A description of what you want to analysis.
exp	Optional. A glyexp::experiment() object to provide more context to the LLM.
group_col	The column name of the group variable in the experiment. Default to "group".
model	Model to use. Defaults to getOption("glysmith.ai_model"), or "deepseek-chat" for DeepSeek and the provider default for other providers.
max_retries	Maximum number of retries when the AI output is invalid. Default to 3.
provider	AI provider passed to ellmer. One of "deepseek", "openai", "anthropic", "gemini", "openrouter", or "openai_compatible". Defaults to getOption("glysmith.ai_provider", "deepseek").
api_key	API key for the selected provider. If NULL, the provider specific environment variable is used. Defaults to getOption("glysmith.ai_api_key").
base_url	Optional base URL for custom or OpenAI-compatible endpoints. Defaults to getOption("glysmith.ai_base_url").

Details

LLMs can be unstable. If you get an error, try again with another description. Make sure to examine the returned blueprint carefully to ensure it's what you want. You can also create parallel analysis branches with br("name", step_..., step_...), which will namespace outputs with the branch prefix. If the LLM needs required information to proceed, it may ask clarifying questions interactively and then retry with your answers. After a blueprint is generated, the description is printed and, in interactive sessions, you can press ENTER to accept it or type new requirements to refine the blueprint. This review step can repeat until you accept the plan.

Here are some examples that works:

- "I want to know what pathways are enriched for my differentially expressed glycoforms."
- "I want a heatmap and a pca plot. I have already performed preprocessing myself."
- "I have a glycomics dataset. I want to calculate derived traits and perform DEA on them."

`modify_blueprint` *Modify a Blueprint using Natural Language*

Description

[Experimental] Ask a Large Language Model (LLM) to modify an existing blueprint for glycomics or glycoproteomics data analysis. DeepSeek is used by default for backward compatibility. Other `ellmer` providers can be selected with `provider`, `model`, and provider-specific API key configuration.

Usage

```
modify_blueprint(
  bp,
  description,
  qa_history = NULL,
  exp = NULL,
  group_col = "group",
  model = getOption("glysmith.ai_model", NULL),
  max_retries = 3,
  provider = getOption("glysmith.ai_provider", "deepseek"),
  api_key = getOption("glysmith.ai_api_key", NULL),
  base_url = getOption("glysmith.ai_base_url", NULL)
)
```

Arguments

`bp`	A `glysmith_blueprint` object.
`description`	A description of how you want to modify the blueprint.
`qa_history`	Character vector of Q&A pairs from `inquire_blueprint()`.
`exp`	Optional. A `glyexp::experiment()` object to provide more context to the LLM.
`group_col`	The column name of the group variable in the experiment. Default to "group".
`model`	Model to use. Defaults to `getOption("glysmith.ai_model")`, or "deepseek-chat" for DeepSeek and the provider default for other providers.
`max_retries`	Maximum number of retries when the AI output is invalid. Default to 3.
`provider`	AI provider passed to `ellmer`. One of "deepseek", "openai", "anthropic", "gemini", "openrouter", or "openai_compatible". Defaults to `getOption("glysmith.ai_provider", "deepseek")`.
`api_key`	API key for the selected provider. If `NULL`, the provider specific environment variable is used. Defaults to `getOption("glysmith.ai_api_key")`.
`base_url`	Optional base URL for custom or OpenAI-compatible endpoints. Defaults to `getOption("glysmith.ai_base_url")`.

Details

LLMs can be unstable. If you get an error, try again with another description. Make sure to examine the returned blueprint carefully to ensure it's what you want. This function is a companion of `inquire_blueprint()`. If the LLM needs required information to proceed, it may ask clarifying questions interactively and then retry with your answers.

`polish_report` *Render a HTML Report for GlySmith Results*

Description

Generate a self-contained HTML report for a `glysmith_result` object. The report is rendered via `rmarkdown::render()` using an internal R Markdown template. If `use_ai` is TRUE, the report text will be polished, organized into sections, paired with plot descriptions, and summarized using the configured `ellmer` provider. DeepSeek is used by default for backward compatibility.

Usage

```
polish_report(
  x,
  output_file,
  title = "GlySmith report",
  open = interactive(),
  use_ai = FALSE,
  ai_provider = getOption("glysmith.ai_provider", "deepseek"),
  ai_model = getOption("glysmith.ai_model", NULL),
  ai_api_key = getOption("glysmith.ai_api_key", NULL),
  ai_base_url = getOption("glysmith.ai_base_url", NULL)
)
```

Arguments

`x`	A `glysmith_result` object.
`output_file`	Path to the output HTML file.
`title`	Report title.
`open`	Whether to open the report in a browser after rendering.
`use_ai`	Whether to polish the report text, organize sections, generate plot descriptions, and add a summary using AI with the configured `ellmer` provider. Default is FALSE.
`ai_provider`	AI provider passed to `ellmer` when `use_ai = TRUE`. One of "deepseek", "openai", "anthropic", "gemini", "openrouter", or "openai_compatible". Defaults to `getOption("glysmith.ai_provider", "deepseek")`.
`ai_model`	AI model to use when `use_ai = TRUE`. Defaults to `getOption("glysmith.ai_model")`, or "deepseek-chat" for DeepSeek and the provider default for other providers.

ai_api_key	API key for the selected provider. If NULL, the provider specific environment variable is used. Defaults to getOption("glysmith.ai_api_key").
ai_base_url	Optional base URL for custom or OpenAI-compatible endpoints. Defaults to getOption("glysmith.ai_base_url").

Value

The normalized path to the generated HTML file.

Examples

```
## Not run:
library(glyexp)
exp <- real_experiment2
result <- forge_analysis(exp)
polish_report(result, tempfile(fileext = ".html"), open = FALSE)

## End(Not run)
```

---

quench_result *Save GlySmith Result*

Description

Save processed experiment, plots and tables of a glysmith result object to a directory. A README.md file will also be generated to describe the saved outputs.

Usage

```
quench_result(
  x,
  dir,
  plot_ext = "pdf",
  table_ext = "csv",
  plot_width = 5,
  plot_height = 5
)
```

Arguments

x	A glysmith result object.
dir	The directory to save the result.
plot_ext	The extension of the plot files. Either "pdf", "png" or "svg". Default is "pdf".
table_ext	The extension of the table files. Either "csv" or "tsv". Default is "csv".
plot_width	The width of the plot in inches. Default is 5.
plot_height	The height of the plot in inches. Default is 5.

Examples

```
## Not run:
library(glyexp)
exp <- real_experiment2
result <- forge_analysis(exp)
quench_result(result, tempdir())

## End(Not run)
```

`step_adjust_protein` *Step: Adjust Protein Abundance*

Description

Adjust glycoform quantification values by correcting for protein abundance utilizing `glyclean::adjust_protein()`. Usually this step should be run after `step_preprocess()`.

This step requires `exp` (experiment data).

Usage

```
step_adjust_protein(pro_expr_path = NULL, method = "ratio")
```

Arguments

`pro_expr_path`	Path to the protein expression matrix file. If `NULL`, this step will be skipped. Can be: • A CSV/TSV file with the first column as protein accessions and remaining columns as sample names. • An RDS file with a matrix or data.frame with row names as protein accessions and columns as sample names.
`method`	The method to use for protein adjustment. Either "ratio" or "reg". Default is "ratio".

Details

Data required:

- `exp`: The experiment to adjust

Data generated:

- `unadj_exp`: The original experiment (previous `exp`, saved for reference)

This step is special in that it silently overwrites the `exp` data with the adjusted experiment. This ensures that no matter if adjustment is performed or not, the "active" experiment is always under the key `exp`. The previous `exp` is saved as `unadj_exp` for reference.

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Include this step only if the user explicitly asks for protein adjustment.
- If protein adjustment is needed and the pro_expr_path is not provided, ask for it and explain how to prepare the file:
  - CSV/TSV: first column is protein accessions; remaining columns are sample names.
  - RDS: a matrix/data.frame with row names as protein accessions and columns as sample names.
- You MUST provide a detailed explanation of how to prepare the file.
- With out the file, the step is invalid.

See Also

glyclean::adjust_protein()

Examples

```
fake_pro_expr_mat <- matrix(rnorm(100), nrow = 10, ncol = 10)
rownames(fake_pro_expr_mat) <- paste0("P", seq_len(10))
colnames(fake_pro_expr_mat) <- paste0("S", seq_len(10))
fake_pro_expr_path <- tempfile(fileext = ".rds")
saveRDS(fake_pro_expr_mat, fake_pro_expr_path)
step_adjust_protein(fake_pro_expr_path)
```

---

step_correlation *Step: Correlation Analysis*

Description

Perform pairwise correlation analysis using glystats::gly_cor() and visualize the correlation matrix using glyvis::plot_corrplot(). This step calculates correlation coefficients and p-values for all pairs of variables or samples.

This step depends on the on parameter (default: exp).

- When on = "exp", requires exp (usually after step_preprocess()).
- When on = "sig_exp", requires sig_exp from one of step_dea_limma(), step_dea_ttest(), step_dea_wilcox(), step_dea_anova(), or step_dea_kruskal().
- When on = "trait_exp", requires trait_exp from step_derive_traits().
- When on = "sig_trait_exp", requires sig_trait_exp from DEA on traits.

- When on = "dynamic_motif_exp", requires dynamic_motif_exp from step_quantify_dynamic_motifs().
- When on = "sig_dynamic_motif_exp", requires sig_dynamic_motif_exp from DEA on motifs.
- When on = "branch_motif_exp", requires branch_motif_exp from step_quantify_branch_motifs().
- When on = "sig_branch_motif_exp", requires sig_branch_motif_exp from DEA on motifs.

Usage

```
step_correlation(
  on = "exp",
  on_cor = c("variable", "sample"),
  method = c("pearson", "spearman"),
  p_adj_method = "BH",
  plot_width = 7,
  plot_height = 7,
  ...
)
```

Arguments

on	Name of the experiment to run correlation analysis on. Can be "exp", "sig_exp", "trait_exp", "sig_trait_exp", "dynamic_motif_exp", "sig_dynamic_motif_exp", "branch_motif_exp", "sig_branch_motif_exp".
on_cor	A character string specifying what to correlate. Either "variable" (default) to correlate variables/features, or "sample" to correlate samples.
method	A character string indicating which correlation coefficient is to be computed. One of "pearson" (default) or "spearman".
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.
plot_width	Width of the plot in inches. Default is 7.
plot_height	Height of the plot in inches. Default is 7.
...	Additional arguments passed to glystats::gly_cor().

Details

Data required:

- exp (if on = "exp"): The experiment to run correlation analysis on
- trait_exp (if on = "trait_exp"): The trait experiment to run correlation analysis on
- dynamic_motif_exp (if on = "dynamic_motif_exp"): The dynamic motif experiment to run correlation analysis on
- branch_motif_exp (if on = "branch_motif_exp"): The branch motif experiment to run correlation analysis on

Tables generated (with suffixes):

- `correlation`: A table containing pairwise correlation results with columns:
  - `variable1`, `variable2` (or `sample1`, `sample2` if `on = "sample"`)
  - `cor`: Correlation coefficient
  - `p_val`: P-value from correlation test
  - `p_adj`: Adjusted p-value (if p_adj_method is not NULL)

Plots generated (with suffixes):

- `correlation`: A correlation matrix heatmap

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step to explore relationships between variables or samples.
- Be careful to use when sample size or variable number is large (> 50). Before using this step for large data, ask the user if they want to proceed.

See Also

`glystats::gly_cor()`, `glyvis::plot_corrplot()`

Examples

```
step_correlation()
step_correlation(on = "sig_exp")
step_correlation(on_cor = "sample", method = "spearman")
```

---

step_cox *Step: Cox Proportional Hazards Model*

Description

Perform survival analysis by fitting a Cox proportional hazards model using `glystats::gly_cox()` for each variable. This step identifies variables associated with survival outcomes.

This step depends on the on parameter (default: `exp`).

- When `on = "exp"`, requires `exp` (usually after `step_preprocess()`).
- When `on = "sig_exp"`, requires `sig_exp` from one of `step_dea_limma()`, `step_dea_ttest()`, `step_dea_wilcox()`, `step_dea_anova()`, or `step_dea_kruskal()`.
- When `on = "trait_exp"`, requires `trait_exp` from `step_derive_traits()`.
- When `on = "sig_trait_exp"`, requires `sig_trait_exp` from DEA on traits.
- When `on = "dynamic_motif_exp"`, requires `dynamic_motif_exp` from `step_quantify_dynamic_motifs()`.

- When on = "sig_dynamic_motif_exp", requires sig_dynamic_motif_exp from DEA on motifs.
- When on = "branch_motif_exp", requires branch_motif_exp from step_quantify_branch_motifs().
- When on = "sig_branch_motif_exp", requires sig_branch_motif_exp from DEA on motifs.

Usage

```
step_cox(
  on = "exp",
  time_col = "time",
  event_col = "event",
  p_adj_method = "BH",
  ...
)
```

Arguments

on	Name of the experiment to run Cox regression on. Can be "exp", "sig_exp", "trait_exp", "sig_trait_exp", "dynamic_motif_exp", "sig_dynamic_motif_exp", "branch_motif_exp", "sig_branch_motif_exp".
time_col	Column name in sample information containing survival time. Default is "time".
event_col	Column name in sample information containing event indicator (1 for event, 0 for censoring). Default is "event".
p_adj_method	Method for adjusting p-values. See p.adjust.methods. Default is "BH". If NULL, no adjustment is performed.
...	Additional arguments passed to glystats::gly_cox().

Details

Data required:

- exp (if on = "exp"): The experiment to run Cox regression on
- trait_exp (if on = "trait_exp"): The trait experiment to run Cox regression on
- dynamic_motif_exp (if on = "dynamic_motif_exp"): The dynamic motif experiment to run Cox regression on
- branch_motif_exp (if on = "branch_motif_exp"): The branch motif experiment to run Cox regression on

The experiment must contain survival data with time_col and event_col columns in the sample information.

Tables generated (with suffixes):

- cox: A table containing Cox regression results with 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)

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step when users want to identify variables associated with survival outcomes.
- This step requires survival data (time and event columns) in the sample information.
- Always ask for the column names for survival data, unless explicitly provided.

See Also

`glystats::gly_cox()`, `survival::coxph()`

Examples

```
step_cox()
step_cox(time_col = "survival_time", event_col = "death")
step_cox(on = "sig_exp", p_adj_method = "bonferroni")
```

`step_dea_anova` *Step: Differential Expression Analysis (DEA) using ANOVA*

Description

Run differential analysis using ANOVA via `glystats::gly_anova()`, then filter the experiment to keep only the differentially expressed variables using `glystats::filter_sig_vars()`. By default, this runs DEA on the main experiment (`exp`), but can be configured to run on derived traits (`trait_exp`) or other experiment objects.

This step depends on the `on` parameter (default: `exp`).

- When `on = "exp"`, requires `exp` (usually after `step_preprocess()`).
- When `on = "trait_exp"`, requires `trait_exp` from `step_derive_traits()`.
- When `on = "dynamic_motif_exp"`, requires `dynamic_motif_exp` from `step_quantify_dynamic_motifs()`.
- When `on = "branch_motif_exp"`, requires `branch_motif_exp` from `step_quantify_branch_motifs()`.

Usage

```
step_dea_anova(
  on = "exp",
  p_adj_method = "BH",
  filter_p_adj_cutoff = 0.05,
  filter_p_val_cutoff = NULL,
  filter_fc_cutoff = NULL,
  filter_on = "main_test",
  filter_comparison = NULL,
  ...
)
```

Arguments

on	Name of the experiment data in ctx$data to run analysis on. Default is "exp" for differential expression analysis. Use "trait_exp" for differential trait analysis. Use "dynamic_motif_exp" for differential dynamic motif analysis. Use "branch_motif_exp" for differential branch motif analysis.
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.
filter_p_adj_cutoff	Adjusted p-value cutoff for filtering.
filter_p_val_cutoff	Raw p-value cutoff for filtering.
filter_fc_cutoff	Fold change cutoff for filtering.
filter_on	Name of the test to filter on. Default is "main_test". Can also be "post_hoc_test".
filter_comparison	Name of the comparison to filter on.
...	Additional arguments passed to stats::aov().

Details

Data required:

- Depends on on parameter (default: exp)

Data generated:

- dea_res: The DEA results (if on = "exp", default)
- dta_res: The DTA results (if on = "trait_exp")
- dynamic_dma_res: The DMA results (if on = "dynamic_motif_exp")
- branch_dma_res: The DMA results (if on = "branch_motif_exp")
- sig_exp: The filtered experiment (if on = "exp", default)
- sig_trait_exp: The filtered trait experiment (if on = "trait_exp")
- sig_dynamic_motif_exp: The filtered dynamic motif experiment (if on = "dynamic_motif_exp")

- sig_branch_motif_exp: The filtered branch motif experiment (if on = "branch_motif_exp")

Tables generated:

- dea_main_test, dea_post_hoc_test: Tables containing the results (if on = "exp", default)
- dta_main_test, dta_post_hoc_test: Tables containing the results (if on = "trait_exp")
- dynamic_dma_main_test, dynamic_dma_post_hoc_test: Tables containing the results (if on = "dynamic_motif_exp")
- branch_dma_main_test, branch_dma_post_hoc_test: Tables containing the results (if on = "branch_motif_exp")

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Include this step only if the user explicitly asks for ANOVA.

See Also

glystats::gly_anova()

Examples

```
step_dea_anova()
step_dea_anova(on = "trait_exp")  # Differential trait analysis
```

---

step_dea_kruskal *Step: Differential Expression Analysis (DEA) using Kruskal-Wallis test*

Description

Run differential analysis using Kruskal-Wallis analysis via glystats::gly_kruskal(), then filter the experiment to keep only the differentially expressed variables using glystats::filter_sig_vars(). By default, this runs DEA on the main experiment (exp), but can be configured to run on derived traits (trait_exp) or other experiment objects.

This step depends on the on parameter (default: exp).

- When on = "exp", requires exp (usually after step_preprocess()).
- When on = "trait_exp", requires trait_exp from step_derive_traits().
- When on = "dynamic_motif_exp", requires dynamic_motif_exp from step_quantify_dynamic_motifs().
- When on = "branch_motif_exp", requires branch_motif_exp from step_quantify_branch_motifs().

Usage

```
step_dea_kruskal(
  on = "exp",
  p_adj_method = "BH",
  filter_p_adj_cutoff = 0.05,
  filter_p_val_cutoff = NULL,
  filter_fc_cutoff = NULL,
  filter_on = "main_test",
  filter_comparison = NULL,
  ...
)
```

Arguments

on	Name of the experiment data in ctx$data to run analysis on. Default is "exp" for differential expression analysis. Use "trait_exp" for differential trait analysis. Use "dynamic_motif_exp" for differential dynamic motif analysis. Use "branch_motif_exp" for differential branch motif analysis.
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.
filter_p_adj_cutoff	Adjusted p-value cutoff for filtering.
filter_p_val_cutoff	Raw p-value cutoff for filtering.
filter_fc_cutoff	Fold change cutoff for filtering.
filter_on	Filter on "main_test" or "post_hoc_test" for Kruskal-Wallis results.
filter_comparison	Comparison name for post-hoc filtering.
...	Additional arguments passed to glystats::gly_kruskal().

Details

Data required:

- Depends on on parameter (default: exp)

Data generated:

- dea_res: The DEA results (if on = "exp", default)
- dta_res: The DTA results (if on = "trait_exp")
- dynamic_dma_res: The DMA results (if on = "dynamic_motif_exp")
- branch_dma_res: The DMA results (if on = "branch_motif_exp")
- sig_exp: The filtered experiment (if on = "exp", default)
- sig_trait_exp: The filtered trait experiment (if on = "trait_exp")
- sig_dynamic_motif_exp: The filtered dynamic motif experiment (if on = "dynamic_motif_exp")

- `sig_branch_motif_exp`: The filtered branch motif experiment (if `on = "branch_motif_exp"`)

Tables generated:

- `dea_main_test`, `dea_post_hoc_test`: Tables containing the results (if `on = "exp"`, default)
- `dta_main_test`, `dta_post_hoc_test`: Tables containing the results (if `on = "trait_exp"`)
- `dynamic_dma_main_test`, `dynamic_dma_post_hoc_test`: Tables containing the results (if `on = "dynamic_motif_exp"`)
- `branch_dma_main_test`, `branch_dma_post_hoc_test`: Tables containing the results (if `on = "branch_motif_exp"`)

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step only if the user explicitly asks for Kruskal-Wallis test.

See Also

`glystats::gly_kruskal()`

Examples

```
step_dea_kruskal()
step_dea_kruskal(on = "trait_exp")  # Differential trait analysis
```

`step_dea_limma` *Step: Differential Expression Analysis (DEA) using Limma*

Description

Run differential analysis using linear model-based analysis via `glystats::gly_limma()`, then filter the experiment to keep only the differentially expressed variables using `glystats::filter_sig_vars()`. By default, this runs DEA on the main experiment (`exp`), but can be configured to run on derived traits (`trait_exp`) or other experiment objects. This step is the recommended DEA method for all experiments, for both two-group and multi-group experiments.

This step depends on the `on` parameter (default: `exp`).

- When `on = "exp"`, requires `exp` (usually after `step_preprocess()`).
- When `on = "trait_exp"`, requires `trait_exp` from `step_derive_traits()`.
- When `on = "dynamic_motif_exp"`, requires `dynamic_motif_exp` from `step_quantify_dynamic_motifs()`.
- When `on = "branch_motif_exp"`, requires `branch_motif_exp` from `step_quantify_branch_motifs()`.

Usage

```
step_dea_limma(
  on = "exp",
  p_adj_method = "BH",
  covariate_cols = NULL,
  subject_col = NULL,
  ref_group = NULL,
  contrasts = NULL,
  filter_p_adj_cutoff = 0.05,
  filter_p_val_cutoff = NULL,
  filter_fc_cutoff = NULL,
  ...
)
```

Arguments

on	Name of the experiment data in ctx$data to run analysis on. Default is "exp" for differential expression analysis. Use "trait_exp" for differential trait analysis. Use "dynamic_motif_exp" for differential dynamic motif analysis. Use "branch_motif_exp" for differential branch motif analysis.
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.
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.
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.
filter_p_adj_cutoff	Adjusted p-value cutoff for filtering.
filter_p_val_cutoff	Raw p-value cutoff for filtering.
filter_fc_cutoff	Fold change cutoff for filtering.
...	Additional arguments passed to limma::lmFit().

Details

Data required:

- exp (if on = "exp"): The experiment to run DEA on
- trait_exp (if on = "trait_exp"): The trait experiment to run DEA on
- dynamic_motif_exp (if on = "dynamic_motif_exp"): The dynamic motif experiment to run DEA on
- branch_motif_exp (if on = "branch_motif_exp"): The branch motif experiment to run DEA on

Data generated:

- dea_res: The DEA (differential expression analysis) results (if on = "exp", default)
- dta_res: The DTA (differential trait analysis) results (if on = "trait_exp")
- dynamic_dma_res: The DMA results (if on = "dynamic_motif_exp")
- branch_dma_res: The DMA results (if on = "branch_motif_exp")
- sig_exp: The filtered experiment (if on = "exp", default)
- sig_trait_exp: The filtered trait experiment (if on = "trait_exp")
- sig_dynamic_motif_exp: The filtered dynamic motif experiment (if on = "dynamic_motif_exp")
- sig_branch_motif_exp: The filtered branch motif experiment (if on = "branch_motif_exp")

Tables generated:

- dea: A table containing the DEA (differential expression analysis) result (if on = "exp", default)
- dta: A table containing the DTA (differential trait analysis) result (if on = "trait_exp")
- dynamic_dma: A table containing the DMA result (if on = "dynamic_motif_exp")
- branch_dma: A table containing the DMA result (if on = "branch_motif_exp")

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Use this step to perform DEA by default, unless the user asks for other methods.

See Also

glystats::gly_limma()

Examples

```
step_dea_limma()
step_dea_limma(on = "trait_exp")  # Differential trait analysis
step_dea_limma(p_adj_method = "BH")
```

step_dea_ttest *Step: Differential Expression Analysis (DEA) using t-test*

Description

Run differential analysis using t-test via `glystats::gly_ttest()`, then filter the experiment to keep only the differentially expressed variables using `glystats::filter_sig_vars()`. By default, this runs DEA on the main experiment (`exp`), but can be configured to run on derived traits (`trait_exp`) or other experiment objects. Only use this method for experiments with 2 groups.

This step depends on the `on` parameter (default: `exp`).

- When `on = "exp"`, requires `exp` (usually after `step_preprocess()`).
- When `on = "trait_exp"`, requires `trait_exp` from `step_derive_traits()`.
- When `on = "dynamic_motif_exp"`, requires `dynamic_motif_exp` from `step_quantify_dynamic_motifs()`.
- When `on = "branch_motif_exp"`, requires `branch_motif_exp` from `step_quantify_branch_motifs()`.

Usage

```
step_dea_ttest(
  on = "exp",
  p_adj_method = "BH",
  ref_group = NULL,
  filter_p_adj_cutoff = 0.05,
  filter_p_val_cutoff = NULL,
  filter_fc_cutoff = NULL,
  ...
)
```

Arguments

`on`	Name of the experiment data in `ctx$data` to run analysis on. Default is `"exp"` for differential expression analysis. Use `"trait_exp"` for differential trait analysis. Use `"dynamic_motif_exp"` for differential dynamic motif analysis. Use `"branch_motif_exp"` for differential branch motif analysis.
`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.
`filter_p_adj_cutoff`	Adjusted p-value cutoff for filtering.
`filter_p_val_cutoff`	Raw p-value cutoff for filtering.
`filter_fc_cutoff`	Fold change cutoff for filtering.
`...`	Additional arguments passed to `stats::t.test()`.

Details

Data required:

- Depends on on parameter (default: exp)

Data generated:

- dea_res: The DEA results (if on = "exp", default)
- dta_res: The DTA results (if on = "trait_exp")
- dynamic_dma_res: The DMA results (if on = "dynamic_motif_exp")
- branch_dma_res: The DMA results (if on = "branch_motif_exp")
- sig_exp: The filtered experiment (if on = "exp", default)
- sig_trait_exp: The filtered trait experiment (if on = "trait_exp")
- sig_dynamic_motif_exp: The filtered dynamic motif experiment (if on = "dynamic_motif_exp")
- sig_branch_motif_exp: The filtered branch motif experiment (if on = "branch_motif_exp")

Tables generated:

- dea: A table containing the DEA result (if on = "exp", default)
- dta: A table containing the DTA result (if on = "trait_exp")
- dynamic_dma: A table containing the DMA result (if on = "dynamic_motif_exp")
- branch_dma: A table containing the DMA result (if on = "branch_motif_exp")

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Include this step only if the user explicitly asks for t-test.
- If the experiment has more than 2 groups but the user wants a specific two-group comparison, ask which two groups to compare and include step_subset_groups(groups = c("A", "B")) before this step.

See Also

glystats::gly_ttest()

Examples

```
step_dea_ttest()
step_dea_ttest(on = "trait_exp")  # Differential trait analysis
```

`step_dea_wilcox` *Step: Differential Expression Analysis (DEA) using Wilcoxon test*

Description

Run differential analysis using Wilcoxon analysis via `glystats::gly_wilcox()`, then filter the experiment to keep only the differentially expressed variables using `glystats::filter_sig_vars()`. By default, this runs DEA on the main experiment (`exp`), but can be configured to run on derived traits (`trait_exp`) or other experiment objects. Only use this method for experiments with 2 groups.

This step depends on the `on` parameter (default: `exp`).

- When `on = "exp"`, requires `exp` (usually after `step_preprocess()`).
- When `on = "trait_exp"`, requires `trait_exp` from `step_derive_traits()`.
- When `on = "dynamic_motif_exp"`, requires `dynamic_motif_exp` from `step_quantify_dynamic_motifs()`.
- When `on = "branch_motif_exp"`, requires `branch_motif_exp` from `step_quantify_branch_motifs()`.

Usage

```
step_dea_wilcox(
  on = "exp",
  p_adj_method = "BH",
  ref_group = NULL,
  filter_p_adj_cutoff = 0.05,
  filter_p_val_cutoff = NULL,
  filter_fc_cutoff = NULL,
  ...
)
```

Arguments

`on`	Name of the experiment data in `ctx$data` to run analysis on. Default is `"exp"` for differential expression analysis. Use `"trait_exp"` for differential trait analysis. Use `"dynamic_motif_exp"` for differential dynamic motif analysis. Use `"branch_motif_exp"` for differential branch motif analysis.
`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.
`filter_p_adj_cutoff`	Adjusted p-value cutoff for filtering.
`filter_p_val_cutoff`	Raw p-value cutoff for filtering.
`filter_fc_cutoff`	Fold change cutoff for filtering.
`...`	Additional arguments passed to `glystats::gly_wilcox()`.

Details

Data required:

- Depends on on parameter (default: `exp`)

Data generated:

- `dea_res`: The DEA results (if on = `"exp"`, default)
- `dta_res`: The DTA results (if on = `"trait_exp"`)
- `dynamic_dma_res`: The DMA results (if on = `"dynamic_motif_exp"`)
- `branch_dma_res`: The DMA results (if on = `"branch_motif_exp"`)
- `sig_exp`: The filtered experiment (if on = `"exp"`, default)
- `sig_trait_exp`: The filtered trait experiment (if on = `"trait_exp"`)
- `sig_dynamic_motif_exp`: The filtered dynamic motif experiment (if on = `"dynamic_motif_exp"`)
- `sig_branch_motif_exp`: The filtered branch motif experiment (if on = `"branch_motif_exp"`)

Tables generated:

- `dea`: A table containing the DEA result (if on = `"exp"`, default)
- `dta`: A table containing the DTA result (if on = `"trait_exp"`)
- `dynamic_dma`: A table containing the DMA result (if on = `"dynamic_motif_exp"`)
- `branch_dma`: A table containing the DMA result (if on = `"branch_motif_exp"`)

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step only if the user explicitly asks for Wilcoxon test.
- If the experiment has more than 2 groups but the user wants a specific two-group comparison, ask which two groups to compare and include `step_subset_groups(groups = c("A", "B"))` before this step.

See Also

`glystats::gly_wilcox()`

Examples

```
step_dea_wilcox()
step_dea_wilcox(on = "trait_exp")  # Differential trait analysis
```

step_derive_traits *Step: Derived Trait Calculation*

Description

Calculate glycan derived traits using `glydet::derive_traits()`. Advanced glycan structure analysis that summarizes structural properties of a glycome or each glycosite. Need glycan structure information.

This step requires `exp` (experiment data).

Usage

```
step_derive_traits(trait_fns = NULL, mp_fns = NULL, mp_cols = NULL)
```

Arguments

`trait_fns`	A named list of derived trait functions created by trait factories. Names of the list are the names of the derived traits. Default is `NULL`, which means all derived traits in `traits_basic()` are calculated.
`mp_fns`	A named list of meta-property functions. This parameter is useful if your trait functions use custom meta-properties other than those in `all_mp_fns()`. Default is `NULL`, which means all meta-properties in `all_mp_fns()` are used.
`mp_cols`	A character vector of column names in the `var_info` tibble to use as meta-properties. If names are provided, they will be used as names of the meta-properties, otherwise the column names will be used. When `mp_cols` is specified, the selected columns overwrite meta-properties introduced by `mp_fns` with the same names, including built-in meta-properties. Default is `NULL`, which means all columns in `var_info` are available as meta-properties by their existing names. In this default mode, meta-properties introduced by `mp_fns` take precedence over `var_info` columns with the same names.

Details

Data required:

- `exp`: The experiment to calculate derived traits for

Data generated:

- `trait_exp`: The experiment with derived traits

Tables generated:

- `derived_traits`: A table containing the derived traits.

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step by default if the experiment has glycan structures.
- After this step, it should be followed by the DEA and visualization steps.

See Also

`glydet::derive_traits()`

Examples

```
step_derive_traits()
```

---

step_heatmap *Step: Heatmap*

Description

Create a heatmap plot using `glyvis::plot_heatmap()`. The heatmap visualizes expression values across samples.

This step depends on the `on` parameter (default: `exp`).

- When `on = "exp"`, requires `exp` (usually after `step_preprocess()`).
- When `on = "sig_exp"`, requires `sig_exp` from one of `step_dea_limma()`, `step_dea_ttest()`, `step_dea_wilcox()`, `step_dea_anova()`, or `step_dea_kruskal()`.
- When `on = "trait_exp"`, requires `trait_exp` from `step_derive_traits()`.
- When `on = "sig_trait_exp"`, requires `sig_trait_exp` from DEA on traits.
- When `on = "dynamic_motif_exp"`, requires `dynamic_motif_exp` from `step_quantify_dynamic_motifs()`.
- When `on = "sig_dynamic_motif_exp"`, requires `sig_dynamic_motif_exp` from DEA on motifs.
- When `on = "branch_motif_exp"`, requires `branch_motif_exp` from `step_quantify_branch_motifs()`.
- When `on = "sig_branch_motif_exp"`, requires `sig_branch_motif_exp` from DEA on motifs.

Usage

```
step_heatmap(on = "exp", plot_width = 7, plot_height = 7, ...)
```

Arguments

`on`	Name of the experiment data in `ctx$data` to plot. One of "exp", "sig_exp", "trait_exp", "sig_trait_exp", "dynamic_motif_exp", "sig_dynamic_motif_exp", "branch_motif_exp", "sig_branch_motif_exp". Default is "exp".
`plot_width`	Width of the plot in inches. Default is 7.
`plot_height`	Height of the plot in inches. Default is 7.
`...`	Additional arguments passed to `glyvis::plot_heatmap()`.

Details

Data required:

- Depends on on parameter (default: `exp`)

Plots generated:

- `heatmap`: A heatmap plot (if on = `"exp"`)
- `sig_heatmap`: A heatmap plot (if on = `"sig_exp"`)
- `trait_heatmap`: A heatmap plot (if on = `"trait_exp"`)
- `sig_trait_heatmap`: A heatmap plot (if on = `"sig_trait_exp"`)
- `dynamic_motif_heatmap`: A heatmap plot (if on = `"dynamic_motif_exp"`)
- `sig_dynamic_motif_heatmap`: A heatmap plot (if on = `"sig_dynamic_motif_exp"`)
- `branch_motif_heatmap`: A heatmap plot (if on = `"branch_motif_exp"`)
- `sig_branch_motif_heatmap`: A heatmap plot (if on = `"sig_branch_motif_exp"`)

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step if needed.
- It is recommended to use this step on significant results (e.g. on = `"sig_exp"`) if available.

See Also

`glyvis::plot_heatmap()`

Examples

```
step_heatmap()
step_heatmap(on = "sig_exp")
step_heatmap(on = "trait_exp")
```

`step_ident_overview` *Step: Identification Overview*

Description

Summarize the experiment using `glyexp::summarize_experiment()`. This is usually the first step, BEFORE `step_preprocess()`. Very light-weight to run, so always include it.

This step requires `exp` (experiment data).

Usage

```
step_ident_overview(count_struct = NULL)
```

Arguments

`count_struct`	For counting glycopeptides and glycoforms. whether to count the number of glycan structures or glycopeptides. If `TRUE`, glycopeptides or glycoforms bearing different glycan structures with the same glycan composition are counted as different ones. If not provided (NULL), defaults to `TRUE` if `glycan_structure` column exists in the variable information tibble, otherwise `FALSE`.

Details

Data required:

- `exp`: The experiment to summarize

Tables generated:

- `summary`: A table containing the identification overview of the experiment

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Always include this step by default unless the user explicitly excludes it.
- Use it as the first step in the blueprint.

See Also

`glyexp::summarize_experiment()`

Examples

```
step_ident_overview()
```

`step_infer_structure` *Step: Infer Glycan Structures*

Description

Infer glycan structures from the `glycan_composition` column in `var_info`. This step uses `glyanno::comp_to_struc()` with a structure database from `glydb::glydb_structures()` and keeps only variables with an inferred structure.

This step requires `exp` (experiment data).

Usage

```
step_infer_structure(species = NULL, structure_level = "topological")
```

Arguments

`species`	Species name used to restrict the glycan structure database. Default is `NULL`, which does not restrict by species.
`structure_level`	Structure level passed to `glydb::glydb_structures()`. One of `"intact"`, `"topological"`, or `"basic"`. Default is `"topological"`.

Details

Data required:

- `exp`: The experiment whose glycan structures should be inferred

Data generated:

- `uninferred_exp`: The original experiment before structure inference

Tables generated:

- `inferred_structures`: A table containing the inferred structure for each original variable and whether inference succeeded.

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step when the user requests structure-aware analysis but the experiment has glycan compositions and no glycan structures.
- This step should be placed before `step_derive_traits()`, `step_quantify_dynamic_motifs()`, or `step_quantify_branch_motifs()`.

- Mention that variables without inferred structures are removed.
- Always ask for species restriction to improve inference accuracy, but allow users to skip it if they want.

See Also

`glyanno::comp_to_struc()`, `glydb::glydb_structures()`

Examples

```
step_infer_structure()
step_infer_structure(species = "Homo sapiens")
```

`step_logo` *Step: Logo Plot*

Description

Create a logo plot for glycosylation sites using `glyvis::plot_logo()`. The logo plot visualizes the amino acid sequence patterns around glycosylation sites. This step is only applicable for glycoproteomics experiments.

This step depends on the `on` parameter (default: `exp`).

- When `on = "exp"`, requires `exp` (experiment data).
- When `on = "sig_exp"`, requires `sig_exp` from one of `step_dea_limma()`, `step_dea_ttest()`, `step_dea_wilcox()`, `step_dea_anova()`, or `step_dea_kruskal()`.

Usage

```
step_logo(
  on = "exp",
  n_aa = 5L,
  fasta = NULL,
  plot_width = 5,
  plot_height = 3,
  ...
)
```

Arguments

`on`	Name of the experiment data in `ctx$data` to plot. One of "exp", "sig_exp". Default is "exp".
`n_aa`	The number of amino acids to the left and right of the glycosylation site. For example, if `n_aa = 5`, the resulting sequence will contain 11 amino acids.

fasta	The path to the FASTA file containing protein sequences. If glyclean::add_site_seq() has been called on the experiment, this argument can be omitted. When site_sequence is missing and fasta is NULL, UniProt.ws is used to fetch protein sequences automatically.
plot_width	Width of the plot in inches. Default is 5.
plot_height	Height of the plot in inches. Default is 3.
...	Additional arguments passed to ggseqlogo::ggseqlogo().

Details

Data required:

- Depends on on parameter (default: exp)

Plots generated:

- logo: A logo plot (if on = "exp")
- sig_logo: A logo plot (if on = "sig_exp")

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Include this step if the user explicitly asks for logo plot.
- If used, ask user if a FASTA file is provided. Tell the user that if not, protein sequences will be fetched from Uniprot automatically.

See Also

glyvis::plot_logo()

Examples

```
step_logo()
step_logo(fasta = "proteins.fasta")
step_logo(on = "sig_exp")
```

`step_oplsda` *Step: Orthogonal Partial Least Squares Discriminant Analysis (OPLS-DA)*

Description

Perform OPLS-DA using `glystats::gly_oplsda()` and plot it with `glyvis::plot_oplsda()`. OPLS-DA separates variation into predictive (related to group) and orthogonal (unrelated) components. This step only works with binary classification (exactly 2 groups).

This step depends on the `on` parameter (default: `exp`).

- When `on = "exp"`, requires `exp` (usually after `step_preprocess()`).
- When `on = "sig_exp"`, requires `sig_exp` from one of `step_dea_limma()`, `step_dea_ttest()`, `step_dea_wilcox()`, `step_dea_anova()`, or `step_dea_kruskal()`.
- When `on = "trait_exp"`, requires `trait_exp` from `step_derive_traits()`.
- When `on = "sig_trait_exp"`, requires `sig_trait_exp` from DEA on traits.
- When `on = "dynamic_motif_exp"`, requires `dynamic_motif_exp` from `step_quantify_dynamic_motifs()`.
- When `on = "sig_dynamic_motif_exp"`, requires `sig_dynamic_motif_exp` from DEA on motifs.
- When `on = "branch_motif_exp"`, requires `branch_motif_exp` from `step_quantify_branch_motifs()`.
- When `on = "sig_branch_motif_exp"`, requires `sig_branch_motif_exp` from DEA on motifs.

Usage

```
step_oplsda(
  on = "exp",
  pred_i = 1,
  ortho_i = NA,
  scale = TRUE,
  plot_width = 5,
  plot_height = 5,
  ...
)
```

Arguments

`on`	Name of the experiment to run OPLS-DA on. Can be "exp", "sig_exp", "trait_exp", "sig_trait_exp", "dynamic_motif_exp", "sig_dynamic_motif_exp", "branch_motif_exp", "sig_branch_motif_exp".
`pred_i`	Number of predictive components to include. Default is 1.
`ortho_i`	Number of orthogonal components to include. Default is NA (automatic).
`scale`	Logical indicating whether to scale the data. Default is TRUE.
`plot_width`	Width of plots in inches. Default is 5.
`plot_height`	Height of plots in inches. Default is 5.
`...`	Additional arguments passed to `glystats::gly_oplsda()`.

Details

Data required:

- `exp` (if `on = "exp"`): The experiment to run OPLS-DA on
- `trait_exp` (if `on = "trait_exp"`): The trait experiment to run OPLS-DA on
- `dynamic_motif_exp` (if `on = "dynamic_motif_exp"`): The dynamic motif experiment to run OPLS-DA on
- `branch_motif_exp` (if `on = "branch_motif_exp"`): The branch motif experiment to run OPLS-DA on

Tables generated (with suffixes):

- `oplsda_samples`: A table containing the OPLS-DA scores for each sample
- `oplsda_variables`: A table containing the OPLS-DA loadings for each variable
- `oplsda_variance`: A table containing the explained variance for each component
- `oplsda_vip`: A table containing the Variable Importance in Projection (VIP) scores
- `oplsda_perm_test`: A table containing permutation test results

Plots generated (with suffixes):

- `oplsda_scores`: An OPLS-DA score plot colored by group
- `oplsda_loadings`: An OPLS-DA loading plot
- `oplsda_variance`: An OPLS-DA variance (scree) plot
- `oplsda_vip`: An OPLS-DA VIP score plot

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step when users explicitly asks for OPLS-DA.
- This step only works with binary classification (exactly 2 groups). If multiple groups are found, ask if `step_subset_groups()` should be run first.

See Also

`glystats::gly_oplsda()`, `glyvis::plot_oplsda()`

Examples

```
step_oplsda()
step_oplsda(pred_i = 1, ortho_i = 1)
```

step_pca *Step: Principal Component Analysis (PCA)*

Description

Run PCA using `glystats::gly_pca()` and plot it with `glyvis::plot_pca()`. Loading plot for glycoproteomics data can be crowded with too many variables. Ignore the resulting plot if it is not informative.

This step depends on the `on` parameter (default: `exp`).

- When `on = "exp"`, requires `exp` (usually after `step_preprocess()`).
- When `on = "sig_exp"`, requires `sig_exp` from one of `step_dea_limma()`, `step_dea_ttest()`, `step_dea_wilcox()`, `step_dea_anova()`, or `step_dea_kruskal()`.
- When `on = "trait_exp"`, requires `trait_exp` from `step_derive_traits()`.
- When `on = "sig_trait_exp"`, requires `sig_trait_exp` from DEA on traits.
- When `on = "dynamic_motif_exp"`, requires `dynamic_motif_exp` from `step_quantify_dynamic_motifs()`.
- When `on = "sig_dynamic_motif_exp"`, requires `sig_dynamic_motif_exp` from DEA on motifs.
- When `on = "branch_motif_exp"`, requires `branch_motif_exp` from `step_quantify_branch_motifs()`.
- When `on = "sig_branch_motif_exp"`, requires `sig_branch_motif_exp` from DEA on motifs.

Usage

```
step_pca(
  on = "exp",
  center = TRUE,
  scale = TRUE,
  loadings = FALSE,
  screeplot = TRUE,
  plot_width = 5,
  plot_height = 5,
  ...
)
```

Arguments

`on`	Name of the experiment to run PCA on. Can be "exp", "sig_exp", "trait_exp", "sig_trait_exp", "dynamic_motif_exp", "sig_dynamic_motif_exp", "branch_motif_exp", "sig_branch_motif_exp".
`center`	A logical indicating whether to center the data. Default is TRUE.
`scale`	A logical indicating whether to scale the data. Default is TRUE.
`loadings`	Logical indicating whether to generate the loading plot. Default is `FALSE` since loading plots for glycoproteomics data can be crowded.

screeplot	Logical indicating whether to generate the screeplot. Default is TRUE.
plot_width	Width of plots in inches. Default is 5.
plot_height	Height of plots in inches. Default is 5.
...	Additional arguments passed to prcomp().

Details

Data required:

- exp (if on = "exp"): The experiment to run PCA on
- trait_exp (if on = "trait_exp"): The trait experiment to run PCA on
- dynamic_motif_exp (if on = "dynamic_motif_exp"): The dynamic motif experiment to run PCA on
- branch_motif_exp (if on = "branch_motif_exp"): The branch motif experiment to run PCA on

Tables generated (with suffixes):

- pca_samples: A table containing the PCA scores for each sample
- pca_variables: A table containing the PCA loadings for each variable
- pca_eigenvalues: A table containing the PCA eigenvalues

Plots generated (with suffixes):

- pca_scores: A PCA score plot colored by group (always generated)
- pca_loadings: A PCA loading plot (if loadings = TRUE)
- pca_screeplot: A PCA screeplot (if screeplot = TRUE)

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Include this step if needed.

See Also

glystats::gly_pca(), glyvis::plot_pca()

Examples

```
step_pca()
```

`step_plot_qc` *Step: Plot QC*

Description

Generate quality control plots for the experiment using `glyclean` plotting functions. This step can be used before AND after `step_preprocess()` to generate QC plots at different stages.

This step requires `exp` (experiment data).

Usage

```
step_plot_qc(
  when = "post",
  batch_col = "batch",
  rep_col = NULL,
  plot_width = 7,
  plot_height = 5
)
```

Arguments

`when`	Character string indicating when this QC step is run. Use `"pre"` for pre-preprocessing QC, `"post"` for post-preprocessing QC, or any other value for generic QC. This is appended to the step ID. Default is `"post"`.
`batch_col`	Column name for batch information (for `glyclean::plot_batch_pca()`).
`rep_col`	Column name for replicate information (for `glyclean::plot_rep_scatter()`).
`plot_width`	Width of plots in inches. Default is 7.
`plot_height`	Height of plots in inches. Default is 5.

Details

Data required:

- `exp`: The experiment to plot QC for

Plots generated:

- `qc_missing_heatmap`: Missing value heatmap
- `qc_missing_samples_bar`: Missing value bar plot on samples
- `qc_missing_variables_bar`: Missing value bar plot on variables
- `qc_tic_bar`: Total intensity count bar plot
- `qc_rank_abundance`: Rank abundance plot
- `qc_int_boxplot`: Intensity boxplot
- `qc_rle`: RLE plot

- qc_cv_dent: CV density plot
- qc_batch_pca: PCA score plot colored by batch (if batch_col provided)
- qc_rep_scatter: Replicate scatter plots (if rep_col provided)

When when = "pre", plots are prefixed with qc_pre_ to distinguish from post-QC plots. When when = "post" or NULL, plots use the standard qc_ prefix.

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- By default, include this step ONLY after step_preprocess().
- You MUST provide the when parameter to specify when the QC is being run.

See Also

glyclean::plot_missing_heatmap(), glyclean::plot_tic_bar(), and other glyclean plotting functions.

Examples

```
step_plot_qc(when = "pre")
step_plot_qc(when = "post")
```

---

step_plsda *Step: Partial Least Squares Discriminant Analysis (PLS-DA)*

Description

Perform PLS-DA using glystats::gly_plsda() and plot it with glyvis::plot_plsda(). PLS-DA is a supervised method that finds components maximizing covariance between predictors and the response variable (group membership).

This step depends on the on parameter (default: exp).

- When on = "exp", requires exp (usually after step_preprocess()).
- When on = "sig_exp", requires sig_exp from one of step_dea_limma(), step_dea_ttest(), step_dea_wilcox(), step_dea_anova(), or step_dea_kruskal().
- When on = "trait_exp", requires trait_exp from step_derive_traits().
- When on = "sig_trait_exp", requires sig_trait_exp from DEA on traits.
- When on = "dynamic_motif_exp", requires dynamic_motif_exp from step_quantify_dynamic_motifs().
- When on = "sig_dynamic_motif_exp", requires sig_dynamic_motif_exp from DEA on motifs.
- When on = "branch_motif_exp", requires branch_motif_exp from step_quantify_branch_motifs().
- When on = "sig_branch_motif_exp", requires sig_branch_motif_exp from DEA on motifs.

Usage

```
step_plsda(
  on = "exp",
  ncomp = 2,
  scale = TRUE,
  plot_width = 5,
  plot_height = 5,
  ...
)
```

Arguments

on	Name of the experiment to run PLS-DA on. Can be "exp", "sig_exp", "trait_exp", "sig_trait_exp", "dynamic_motif_exp", "sig_dynamic_motif_exp", "branch_motif_exp", "sig_branch_motif_exp".
ncomp	Number of components to include. Default is 2.
scale	Logical indicating whether to scale the data. Default is TRUE.
plot_width	Width of plots in inches. Default is 5.
plot_height	Height of plots in inches. Default is 5.
...	Additional arguments passed to `glystats::gly_plsda()`.

Details

Data required:

- `exp` (if `on = "exp"`): The experiment to run PLS-DA on
- `trait_exp` (if `on = "trait_exp"`): The trait experiment to run PLS-DA on
- `dynamic_motif_exp` (if `on = "dynamic_motif_exp"`): The dynamic motif experiment to run PLS-DA on
- `branch_motif_exp` (if `on = "branch_motif_exp"`): The branch motif experiment to run PLS-DA on

Tables generated (with suffixes):

- `plsda_samples`: A table containing the PLS-DA scores for each sample
- `plsda_variables`: A table containing the PLS-DA loadings for each variable
- `plsda_variance`: A table containing the explained variance for each component
- `plsda_vip`: A table containing the Variable Importance in Projection (VIP) scores
- `plsda_perm_test`: A table containing permutation test results

Plots generated (with suffixes):

- `plsda_scores`: A PLS-DA score plot colored by group
- `plsda_loadings`: A PLS-DA loading plot
- `plsda_variance`: A PLS-DA variance (scree) plot
- `plsda_vip`: A PLS-DA VIP score plot

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step when users explicitly asks for PLS-DA.

See Also

`glystats::gly_plsda()`, `glyvis::plot_plsda()`

Examples

```
step_plsda()
step_plsda(ncomp = 3)
```

`step_preprocess` *Step: Preprocessing*

Description

Preprocess the experiment using `glyclean::auto_clean()`, and remove quality control (QC) samples if exist. This step can be omitted if the experiment is already preprocessed.

This step requires `exp` (experiment data).

Usage

```
step_preprocess(
  batch_col = "batch",
  qc_name = "QC",
  normalize_to_try = NULL,
  impute_to_try = NULL,
  remove_preset = "discovery",
  batch_prop_threshold = 0.3,
  check_batch_confounding = TRUE,
  batch_confounding_threshold = 0.4,
  rep_col = NULL
)
```

Arguments

batch_col	Column name for batch information (for QC plots and batch effect handling).
qc_name	Name of QC sample group (used for QC sample detection in preprocessing).
normalize_to_try	Normalization methods to try during auto_clean.
impute_to_try	Imputation methods to try during auto_clean.
remove_preset	Preset for data removal: "discovery", "biomarker", or NULL.
batch_prop_threshold	Threshold for batch proportion filtering.
check_batch_confounding	Whether to check for batch confounding.
batch_confounding_threshold	Threshold for batch confounding detection.
rep_col	Column name for replicate information (for QC plots).

Details

Data required:

- exp: The experiment to preprocess

Data generated:

- raw_exp: The raw experiment (previous exp, saved for reference)

This step is special in that it silently overwrites the exp data with the preprocessed experiment. This ensures that no matter if preprocessing is performed or not, the "active" experiment is always under the key exp. The previous exp is saved as raw_exp for reference.

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Always include this step by default unless the user explicitly excludes it or tell you she/he has already performed preprocessing.
- Ask for the column name for batch information if not provided.
- Ask for QC samples in the experiment if not provided. If so, ask the group name of the QC samples. Explain to the user that if it is "QC" for example, the samples with "QC" in the group_col column will be considered as QC samples. And these QC samples will be used for choosing the best normalization and imputation methods. Also mention that QC samples will be excluded after preprocessing.
- If the user intents to perform biomarker related analysis, set remove_preset to "biomarker".
- Use default values for other arguments unless the user explicitly specifies otherwise.

See Also

glyclean::auto_clean()

Examples

```
step_preprocess()
step_preprocess(remove_preset = "discovery")
```

`step_quantify_branch_motifs` *Step: Quantify Branch Motifs*

Description

Quantify N-glycan branch motifs using `glydet::quantify_motifs()` with `glymotif::branch_motifs()`. This extracts specific N-glycan branching patterns (bi-antennary, tri-antennary, etc.). Only works with N-glycans.

This step requires `exp` (experiment data).

Usage

```
step_quantify_branch_motifs(method = "relative")
```

Arguments

`method`	Method for motif quantification ("relative" or "absolute"). Default is "relative".

Details

Data required:

- `exp`: The experiment to quantify motifs for (must be N-glycans)

Data generated:

- `branch_motif_exp`: The experiment with quantified branch motifs

Tables generated:

- `branch_motifs`: A table containing the quantified branch motifs.

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step if motif analysis is needed specifically for N-glycans.
- This step should be followed by DEA and visualization steps.

See Also

glydet::quantify_motifs(), glymotif::branch_motifs()

Examples

```
step_quantify_branch_motifs()
```

`step_quantify_dynamic_motifs`

Step: Quantify Dynamic Motifs

Description

Quantify glycan motifs using `glydet::quantify_motifs()` with `glymotif::dynamic_motifs()`. This extracts all possible motifs from glycan structures. Works with any glycan type.

This step requires `exp` (experiment data).

Usage

```
step_quantify_dynamic_motifs(max_size = 3, method = "relative")
```

Arguments

`max_size`	Maximum size of motifs to extract. Default is 3.
`method`	Method for motif quantification ("relative" or "absolute"). Default is "relative".

Details

Data required:

- `exp`: The experiment to quantify motifs for

Data generated:

- `dynamic_motif_exp`: The experiment with quantified motifs

Tables generated:

- `dynamic_motifs`: A table containing the quantified motifs.

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step if motif analysis is needed for non-N-glycans or when comprehensive motif extraction is desired.
- This step should be followed by DEA and visualization steps.

See Also

glydet::quantify_motifs(), glymotif::dynamic_motifs()

Examples

```
step_quantify_dynamic_motifs()
```

---

step_roc *Step: ROC Analysis*

Description

Perform ROC analysis using glystats::gly_roc(), extract top 10 variables with highest AUC, and plot ROC curves for these variables using glyvis::plot_roc().

This step requires exp (experiment data).

Usage

```
step_roc(pos_class = NULL, plot_width = 5, plot_height = 5)
```

Arguments

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.
plot_width	Width of the plot in inches. Default is 5.
plot_height	Height of the plot in inches. Default is 5.

Details

Data required:

- exp: The experiment to perform ROC analysis on

Tables generated:

- roc_auc: A table containing the ROC AUC values for all variables

Plots generated:

- roc_curves: ROC curves for the top 10 variables

Value

A glysmith_step object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step if the user explicitly asks for ROC analysis, or if he/she mentions "biomarker(s)" in the prompt.
- If the experiment has more than 2 groups but the user wants a specific two-group comparison, ask which two groups to compare and include `step_subset_groups(groups = c("A", "B"))` before this step.

See Also

`glystats::gly_roc()`, `glyvis::plot_roc()`

Examples

```
step_roc()
```

---

step_sig_boxplot *Step: Significant Variables Boxplot*

Description

Create boxplots for the most significant variables from DEA analysis using `glyvis::plot_boxplot()`. The function selects the top `n_top` variables with the lowest adjusted p-values from the DEA results and plots their expression values grouped by sample groups.

This step depends on the `on` parameter (default: `sig_exp`).

- When `on = "sig_exp"`, requires `sig_exp` from one of `step_dea_limma()`, `step_dea_ttest()`, `step_dea_wilcox()`, `step_dea_anova()`, or `step_dea_kruskal()`.
- When `on = "sig_trait_exp"`, requires `sig_trait_exp` from DEA on traits.
- When `on = "sig_dynamic_motif_exp"`, requires `sig_dynamic_motif_exp` from DEA on motifs.
- When `on = "sig_branch_motif_exp"`, requires `sig_branch_motif_exp` from DEA on motifs.

The number of variables is limited to a maximum of 25, as enforced by `glyvis::plot_boxplot()`.

Usage

```
step_sig_boxplot(
  on = "sig_exp",
  n_top = 25,
  panel_width = 1.5,
  panel_height = 1.2,
  min_width = 5,
  min_height = 3,
```

```
  max_width = 14,
  max_height = 12,
  ...
)
```

Arguments

on	Name of the experiment data in ctx$data to plot. One of "sig_exp", "sig_trait_exp", "sig_dynamic_motif_exp", "sig_branch_motif_exp". Default is "sig_exp".
n_top	Number of top significant variables to plot. Must be between 1 and 25 (inclusive). Default is 25.
panel_width	Width of each panel in inches. Default is 1.5.
panel_height	Height of each panel in inches. Default is 1.2.
min_width	Minimum plot width in inches. Default is 5.
min_height	Minimum plot height in inches. Default is 3.
max_width	Maximum plot width in inches. Default is 14.
max_height	Maximum plot height in inches. Default is 12.
...	Additional arguments passed to glyvis::plot_boxplot().

Details

Data required:

- Depends on on parameter:
  - sig_exp (default): Significant experiment from DEA
  - sig_trait_exp: Significant trait experiment from DTA
  - sig_dynamic_motif_exp: Significant dynamic motif experiment from DMA
  - sig_branch_motif_exp: Significant branch motif experiment from DMA

Plots generated:

- sig_boxplot: A boxplot of significant variables (if on = "sig_exp")
- sig_trait_boxplot: A boxplot of significant traits (if on = "sig_trait_exp")
- sig_dynamic_motif_boxplot: A boxplot of significant dynamic motifs (if on = "sig_dynamic_motif_exp")
- sig_branch_motif_boxplot: A boxplot of significant branch motifs (if on = "sig_branch_motif_exp")

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Include this step after DEA steps to visualize the significant variables.
- This step is particularly useful for understanding the expression patterns of the most differentially expressed features across groups.

See Also

glyvis::plot_boxplot()

Examples

```
step_sig_boxplot()
step_sig_boxplot(n_top = 12)
step_sig_boxplot(on = "sig_trait_exp")
```

---

step_sig_enrich_do *Step: DO Enrichment Analysis on Differentially Expressed Variables*

Description

Perform Disease Ontology enrichment analysis on differentially expressed variables using glyfun::enrich_ora_do().

This step requires dea_res (differential analysis result from DEA). Run one of step_dea_limma(), step_dea_ttest(), or step_dea_wilcox() before this step. Only execute for glycoproteomics experiments with exactly 2 groups. If used for glycomics experiments, the step will be skipped. Use all genes in OrgDb as the background.

Usage

```
step_sig_enrich_do(
  universe = "all",
  plot_type = "dotplot",
  plot_width = 7,
  plot_height = 7,
  ...
)
```

Arguments

universe	The universe (background) to use for enrichment analysis. One of "all" (all genes in OrgDb), "detected" (detected variables in exp).
plot_type	Plot type for enrichment results ("dotplot", "barplot", etc.).
plot_width	Width of the plot in inches. Default is 7.
plot_height	Height of the plot in inches. Default is 7.
...	Additional arguments passed to glyfun::enrich_ora_do().

Details

Data required:

- exp: The experiment to perform Disease Ontology enrichment analysis for
- dea_res: The result from DEA, generated by step_dea_xxx().

Tables generated:

- do_enrich: A table containing the Disease Ontology enrichment results.

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step if user asks for it.
- Leave `universe` to "all" (by default) unless the user explicitly mentions that the background should be the detected variables in `exp`.
- If the experiment has more than 2 groups but the user wants enrichment for a specific two-group comparison, ask which two groups to compare and include `step_subset_groups(groups = c("A", "B"))` before DEA and enrichment steps.

See Also

`glyfun::enrich_ora_do()`

Examples

```
step_sig_enrich_do()
step_sig_enrich_do(plot_type = "barplot")
```

`step_sig_enrich_go` *Step: GO Enrichment Analysis on Differentially Expressed Variables*

Description

Perform GO enrichment analysis on differentially expressed variables using `glyfun::enrich_ora_go()`.

This step requires `dea_res` (differential analysis result from DEA). Run one of `step_dea_limma()`, `step_dea_ttest()`, or `step_dea_wilcox()` before this step. Only execute for glycoproteomics experiments with exactly 2 groups. If used for glycomics experiments, the step will be skipped. Use all genes in OrgDb as the background.

Usage

```
step_sig_enrich_go(
  universe = "all",
  plot_type = "dotplot",
  plot_width = 7,
  plot_height = 7,
  ...
)
```

Arguments

universe	The universe (background) to use for enrichment analysis. One of "all" (all genes in OrgDb), "detected" (detected variables in exp).
plot_type	Plot type for enrichment results ("dotplot", "barplot", etc.).
plot_width	Width of the plot in inches. Default is 7.
plot_height	Height of the plot in inches. Default is 7.
...	Additional arguments passed to glyfun::enrich_ora_go().

Details

Data required:

- exp: The experiment to perform GO enrichment analysis for
- dea_res: The result from DEA, generated by step_dea_xxx().

Tables generated:

- go_enrich: A table containing the GO enrichment results.

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Include this step by default if DEA is performed on glycoproteomics data.
- Leave universe to "all" (by default) unless the user explicitly mentions that the background should be the detected variables in exp.
- If the experiment has more than 2 groups but the user wants enrichment for a specific two-group comparison, ask which two groups to compare and include step_subset_groups(groups = c("A", "B")) before DEA and enrichment steps.

See Also

glyfun::enrich_ora_go()

Examples

```
step_sig_enrich_go()
step_sig_enrich_go(plot_type = "barplot")
```

`step_sig_enrich_kegg` *Step: KEGG Enrichment Analysis on Differentially Expressed Variables*

Description

Perform KEGG enrichment analysis on differentially expressed variables using `glyfun::enrich_ora_kegg()`.

This step requires `dea_res` (differential analysis result from DEA). Run one of `step_dea_limma()`, `step_dea_ttest()`, or `step_dea_wilcox()` before this step. Only execute for glycoproteomics experiments with exactly 2 groups. If used for glycomics experiments, the step will be skipped. Use all genes in OrgDb as the background.

Usage

```
step_sig_enrich_kegg(
  universe = "all",
  plot_type = "dotplot",
  plot_width = 7,
  plot_height = 7,
  ...
)
```

Arguments

`universe`	The universe (background) to use for enrichment analysis. One of "all" (all genes in OrgDb), "detected" (detected variables in `exp`).
`plot_type`	Plot type for enrichment results ("dotplot", "barplot", etc.).
`plot_width`	Width of the plot in inches. Default is 7.
`plot_height`	Height of the plot in inches. Default is 7.
`...`	Additional arguments passed to `glyfun::enrich_ora_kegg()`.

Details

Data required:

- `exp`: The experiment to perform KEGG enrichment analysis for
- `dea_res`: The result from DEA, generated by `step_dea_xxx()`.

Tables generated:

- `kegg_enrich`: A table containing the KEGG enrichment results.

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step by default if DEA is performed on glycoproteomics data.
- Leave `universe` to "all" (by default) unless the user explicitly mentions that the background should be the detected variables in `exp`.
- If the experiment has more than 2 groups but the user wants enrichment for a specific two-group comparison, ask which two groups to compare and include `step_subset_groups(groups = c("A", "B"))` before DEA and enrichment steps.

See Also

`glyfun::enrich_ora_kegg()`

Examples

```
step_sig_enrich_kegg()
step_sig_enrich_kegg(plot_type = "barplot")
```

---

step_sig_enrich_ncg *Step: NCG Enrichment Analysis on Differentially Expressed Variables*

Description

Perform NCG enrichment analysis on differentially expressed variables using `glyfun::enrich_ora_ncg()`.

This step requires `dea_res` (differential analysis result from DEA). Run one of `step_dea_limma()`, `step_dea_ttest()`, or `step_dea_wilcox()` before this step. Only execute for glycoproteomics experiments with exactly 2 groups. If used for glycomics experiments, the step will be skipped. Use all genes in OrgDb as the background.

Usage

```
step_sig_enrich_ncg(
  universe = "all",
  plot_type = "dotplot",
  plot_width = 7,
  plot_height = 7,
  ...
)
```

Arguments

`universe`	The universe (background) to use for enrichment analysis. One of "all" (all genes in OrgDb), "detected" (detected variables in `exp`).
`plot_type`	Plot type for enrichment results ("dotplot", "barplot", etc.).
`plot_width`	Width of the plot in inches. Default is 7.
`plot_height`	Height of the plot in inches. Default is 7.
`...`	Additional arguments passed to `glyfun::enrich_ora_ncg()`.

Details

Data required:

- exp: The experiment to perform NCG enrichment analysis for
- dea_res: The result from DEA, generated by step_dea_xxx().

Tables generated:

- ncg_enrich: A table containing the NCG enrichment results.

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Include this step if user asks for it.
- Leave universe to "all" (by default) unless the user explicitly mentions that the background should be the detected variables in exp.
- If the experiment has more than 2 groups but the user wants enrichment for a specific two-group comparison, ask which two groups to compare and include step_subset_groups(groups = c("A", "B")) before DEA and enrichment steps.

See Also

glyfun::enrich_ora_ncg()

Examples

```
step_sig_enrich_ncg()
step_sig_enrich_ncg(plot_type = "barplot")
```

---

step_sig_enrich_reactome

Step: Reactome Enrichment Analysis on Differentially Expressed Variables

---

Description

Perform Reactome enrichment analysis on differentially expressed variables using glyfun::enrich_ora_reactome().

This step requires dea_res (differential analysis result from DEA). Run one of step_dea_limma(), step_dea_ttest(), or step_dea_wilcox() before this step. Only execute for glycoproteomics experiments with exactly 2 groups. If used for glycomics experiments, the step will be skipped. Use all genes in OrgDb as the background.

Usage

```
step_sig_enrich_reactome(
  universe = "all",
  plot_type = "dotplot",
  plot_width = 7,
  plot_height = 7,
  ...
)
```

Arguments

universe	The universe (background) to use for enrichment analysis. One of "all" (all genes in OrgDb), "detected" (detected variables in exp).
plot_type	Plot type for enrichment results ("dotplot", "barplot", etc.).
plot_width	Width of the plot in inches. Default is 7.
plot_height	Height of the plot in inches. Default is 7.
...	Additional arguments passed to glyfun::enrich_ora_reactome().

Details

Data required:

- exp: The experiment to perform Reactome enrichment analysis for
- dea_res: The result from DEA, generated by step_dea_xxx().

Tables generated:

- reactome_enrich: A table containing the Reactome enrichment results.

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Include this step if user asks for it.
- Leave universe to "all" (by default) unless the user explicitly mentions that the background should be the detected variables in exp.
- If the experiment has more than 2 groups but the user wants enrichment for a specific two-group comparison, ask which two groups to compare and include step_subset_groups(groups = c("A", "B")) before DEA and enrichment steps.

See Also

glyfun::enrich_ora_reactome()

Examples

```
step_sig_enrich_reactome()
step_sig_enrich_reactome(plot_type = "barplot")
```

`step_sig_enrich_wp` *Step: WikiPathways Enrichment Analysis on Differentially Expressed Variables*

Description

Perform WikiPathways enrichment analysis on differentially expressed variables using `glyfun::enrich_ora_wp()`.

This step requires `dea_res` (differential analysis result from DEA). Run one of `step_dea_limma()`, `step_dea_ttest()`, or `step_dea_wilcox()` before this step. Only execute for glycoproteomics experiments with exactly 2 groups. If used for glycomics experiments, the step will be skipped. Use all genes in OrgDb as the background.

Usage

```
step_sig_enrich_wp(
  universe = "all",
  plot_type = "dotplot",
  plot_width = 7,
  plot_height = 7,
  ...
)
```

Arguments

`universe`	The universe (background) to use for enrichment analysis. One of "all" (all genes in OrgDb), "detected" (detected variables in `exp`).
`plot_type`	Plot type for enrichment results ("dotplot", "barplot", etc.).
`plot_width`	Width of the plot in inches. Default is 7.
`plot_height`	Height of the plot in inches. Default is 7.
`...`	Additional arguments passed to `glyfun::enrich_ora_wp()`.

Details

Data required:

- `exp`: The experiment to perform WikiPathways enrichment analysis for
- `dea_res`: The result from DEA, generated by `step_dea_xxx()`.

Tables generated:

- `wp_enrich`: A table containing the WikiPathways enrichment results.

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Include this step if user asks for it.
- Leave universe to "all" (by default) unless the user explicitly mentions that the background should be the detected variables in exp.
- If the experiment has more than 2 groups but the user wants enrichment for a specific two-group comparison, ask which two groups to compare and include step_subset_groups(groups = c("A", "B")) before DEA and enrichment steps.

See Also

glyfun::enrich_ora_wp()

Examples

```
step_sig_enrich_wp()
step_sig_enrich_wp(plot_type = "barplot")
```

---

step_subset_groups *Step: Subset Groups*

Description

Subset the experiment to specific groups using the group column in sample information. This is useful when downstream steps require exactly two groups for comparison. Usually run after step_preprocess() and before DEA or enrichment steps.

This step requires exp (experiment data).

Usage

```
step_subset_groups(groups = NULL)
```

Arguments

groups	Group names to keep. If NULL, this step will be skipped.

Details

Data required:

- exp: The experiment to subset

Data generated:

- full_exp: The original experiment before subsetting

This step overwrites exp in the context with the subset experiment.

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Use this step when the experiment has more than 2 groups but the user wants a specific two-group comparison.
- Ask the user which two groups to compare, and place this step before DEA and enrichment steps.
- Use the order of the user-provided groups to set factor levels.

Examples

```
step_subset_groups(groups = c("H", "C"))
```

---

step_tsne *Step: t-SNE*

Description

Perform t-SNE analysis using glystats::gly_tsne() and plot a t-SNE plot using glyvis::plot_tsne(). Note that the result of t-SNE largely depends on the perplexity parameter. Usually it's a trial-and-error process to find the best value iteratively. If you are not satisfied with the result, manually call glyvis::plot_tsne() with different perplexity values to find the best one.

This step depends on the on parameter (default: exp).

- When on = "exp", requires exp (usually after step_preprocess()).
- When on = "sig_exp", requires sig_exp from one of step_dea_limma(), step_dea_ttest(), step_dea_wilcox(), step_dea_anova(), or step_dea_kruskal().
- When on = "trait_exp", requires trait_exp from step_derive_traits().
- When on = "sig_trait_exp", requires sig_trait_exp from DEA on traits.
- When on = "dynamic_motif_exp", requires dynamic_motif_exp from step_quantify_dynamic_motifs().

- When on = "sig_dynamic_motif_exp", requires sig_dynamic_motif_exp from DEA on motifs.
- When on = "branch_motif_exp", requires branch_motif_exp from step_quantify_branch_motifs().
- When on = "sig_branch_motif_exp", requires sig_branch_motif_exp from DEA on motifs.

Usage

```
step_tsne(
  on = "exp",
  dims = 2,
  perplexity = 30,
  plot_width = 5,
  plot_height = 5,
  ...
)
```

Arguments

on	Name of the experiment to run t-SNE on. Can be "exp", "sig_exp", "trait_exp", "sig_trait_exp", "dynamic_motif_exp", "sig_dynamic_motif_exp", "branch_motif_exp", "sig_branch_motif_exp".
dims	Number of output dimensions. Default is 2.
perplexity	Perplexity parameter for t-SNE. Default is 30.
plot_width	Width of the plot in inches. Default is 5.
plot_height	Height of the plot in inches. Default is 5.
...	Additional arguments passed to Rtsne::Rtsne().

Details

Data required:

- exp (if on = "exp"): The experiment to perform t-SNE on
- trait_exp (if on = "trait_exp"): The trait experiment to perform t-SNE on
- dynamic_motif_exp (if on = "dynamic_motif_exp"): The dynamic motif experiment to perform t-SNE on
- branch_motif_exp (if on = "branch_motif_exp"): The branch motif experiment to perform t-SNE on

Data generated (with suffixes):

- tsne: The t-SNE result

Plots generated (with suffixes):

- tsne: The t-SNE plot

Value

A glysmith_step object.

AI Prompt

This section is for AI in inquire_blueprint() *only.*

- Include this step only when the user explicitly asks for t-SNE.

See Also

glystats::gly_tsne(), glyvis::plot_tsne()

Examples

```
step_tsne()
step_tsne(perplexity = 30)
```

---

step_umap *Step: UMAP*

Description

Perform UMAP analysis using glystats::gly_umap() and plot a UMAP plot using glyvis::plot_umap(). Note that the result of UMAP largely depends on the n_neighbors parameter. Usually it's a trial-and-error process to find the best value iteratively. If you are not satisfied with the result, manually call glyvis::plot_umap() with different n_neighbors values to find the best one.

This step depends on the on parameter (default: exp).

- When on = "exp", requires exp (usually after step_preprocess()).
- When on = "sig_exp", requires sig_exp from one of step_dea_limma(), step_dea_ttest(), step_dea_wilcox(), step_dea_anova(), or step_dea_kruskal().
- When on = "trait_exp", requires trait_exp from step_derive_traits().
- When on = "sig_trait_exp", requires sig_trait_exp from DEA on traits.
- When on = "dynamic_motif_exp", requires dynamic_motif_exp from step_quantify_dynamic_motifs().
- When on = "sig_dynamic_motif_exp", requires sig_dynamic_motif_exp from DEA on motifs.
- When on = "branch_motif_exp", requires branch_motif_exp from step_quantify_branch_motifs().
- When on = "sig_branch_motif_exp", requires sig_branch_motif_exp from DEA on motifs.

Usage

```
step_umap(
  on = "exp",
  n_neighbors = 15,
  n_components = 2,
  plot_width = 5,
  plot_height = 5,
  ...
)
```

Arguments

on	Name of the experiment to run UMAP on. Can be "exp", "sig_exp", "trait_exp", "sig_trait_exp", "dynamic_motif_exp", "sig_dynamic_motif_exp", "branch_motif_exp", "sig_branch_motif_exp".
n_neighbors	Number of neighbors to consider for each point. Default is 15.
n_components	Number of output dimensions. Default is 2.
plot_width	Width of the plot in inches. Default is 5.
plot_height	Height of the plot in inches. Default is 5.
...	Additional arguments passed to `uwot::umap()`.

Details

Data required:

- `exp` (if `on = "exp"`): The experiment to perform UMAP on
- `trait_exp` (if `on = "trait_exp"`): The trait experiment to perform UMAP on
- `dynamic_motif_exp` (if `on = "dynamic_motif_exp"`): The dynamic motif experiment to perform UMAP on
- `branch_motif_exp` (if `on = "branch_motif_exp"`): The branch motif experiment to perform UMAP on

Data generated (with suffixes):

- `umap`: The UMAP result

Plots generated (with suffixes):

- `umap`: The UMAP plot

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Include this step only when the user explicitly asks for UMAP.

See Also

glystats::gly_umap(), glyvis::plot_umap()

Examples

```
step_umap()
step_umap(n_neighbors = 15)
```

---

step_volcano *Step: Volcano Plot*

Description

Create a volcano plot from DEA results using `glyvis::plot_volcano()`.

This step requires one of the DEA steps to be run:

- step_dea_limma() (multi-group comparison is also supported)
- step_dea_ttest()
- step_dea_wilcox()

Usage

```
step_volcano(
  log2fc_cutoff = 1,
  p_cutoff = 0.05,
  p_col = "p_adj",
  plot_width = 5,
  plot_height = 6,
  ...
)
```

Arguments

log2fc_cutoff	The log2 fold change cutoff. Defaults to 1.
p_cutoff	The p-value cutoff. Defaults to 0.05.
p_col	The column name for p-value. Defaults to "p_adj". Can also be "p_val" (raw p-values without multiple testing correction).
plot_width	Width of the plot in inches. Default is 5.
plot_height	Height of the plot in inches. Default is 6.
...	Other arguments passed to EnhancedVolcano::EnhancedVolcano().

Details

Data required:

- `dea_res`: The DEA results from `glystats::gly_limma()`

Plots generated:

- `volcano`: A volcano plot

Value

A `glysmith_step` object.

AI Prompt

This section is for AI in `inquire_blueprint()` *only.*

- Always include this step by default if DEA is performed, and the DEA method is not ANOVA or Kruskal-Wallis.

See Also

`glyvis::plot_volcano()`

Examples

```
step_volcano()
step_volcano(log2fc_cutoff = 2)
```

`write_blueprint` *Save or Load a Blueprint*

Description

- `write_blueprint()` saves a blueprint to a RDS file.
- `read_blueprint()` loads a blueprint from a RDS file.

Usage

```
write_blueprint(bp, file)

read_blueprint(file)
```

Arguments

`bp`	A `blueprint()`.
`file`	A character string giving the name of the file to save to or load from.

Value

Invisibly returns the blueprint object.

Examples

```
bp <- blueprint(
  step_preprocess(),
  step_pca(),
  step_dea_limma(),
)
write_blueprint(bp, tempfile(fileext = ".rds"))
```

Index