Package: glycoTraitR (via r-universe)

May 22, 2026

Type Package

Title Compute and analyze the glycan structrual traits from GPSM data

Version 1.1.0

Description GlycoTraitR is an R package for analyzing glycoproteomics data, particularly glycopeptide-spectrum matches (GPSMs). It supports results generated by the pGlyco3 and Glyco-Decipher search engines. The package parses glycan structures, computes monosaccharide compositions and structural traits, and performs differential analysis of glycan heterogeneity. It constructs trait-by-PSM matrices stored in a SummarizedExperiment object, supports user-defined structural motifs, and provides visualization utilities for interpreting glycan trait changes.

URL `https://github.com/matsui-lab/glycoTraitR`

License MIT + file LICENSE

Encoding UTF-8

Imports igraph, SummarizedExperiment, pbapply, car, ggplot2, rlang

Suggests knitr, BiocStyle, rmarkdown, markdown, testthat (>= 3.0.0)

biocViews Proteomics, MassSpectrometry, Visualization, Software

RoxygenNote 7.3.3

VignetteBuilder knitr

BugReports `https://github.com/matsui-lab/glycoTraitR/issues`

Depends R (>= 4.5.0)

Config/testthat/edition 3

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

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

Date/Publication 2026-04-28 13:06:19 UTC

RemoteUrl https://github.com/bioc/glycoTraitR

RemoteRef HEAD

RemoteSha 7775ec53cddacbbe60743778af072b181a33b54e

Contents

`analyze_trait_changes` *Differential analysis of glycan traits between experimental groups*

Description

Perform group-wise statistical testing on glycan trait matrices stored in a SummarizedExperiment created by `build_trait_se`. For each glycan trait and each site/protein row, the function compares trait intensities across user-specified experimental groups using Welch's t-test and Levene's variance test.

Usage

```
analyze_trait_changes(trait_se, group_col, group_levels, min_psm = 20)
```

Arguments

`trait_se`	A SummarizedExperiment containing trait matrices (one assay per trait), typically returned by `build_trait_se`.
`group_col`	The column name in ‘colData(trait_se)‘ defining sample group membership.
`group_levels`	Character vector specifying which values of ‘group_col‘ to compare (e.g., ‘c("Control","Treatment")‘).
`min_psm`	Minimum required PSM count per group for statistical testing. Default = 20.

Details

Each assay in ‘trait_se‘ represents a glycan trait matrix. The rows as glycopeptides (site-level) or proteins (protein-level). The columns are GPSM count found in samples. For each trait × feature combination, Extract PSM-level trait intensities for samples belonging to the specified ‘group_levels‘. Exclude traits where either group has fewer than ‘min_psm‘ GPSMs. Exclude all-zero traits (boolean-like traits) Run Welch two-sample t-test (‘t.test‘) and Levene's variance test (‘car::leveneTest‘, median centered). A result is returned only if either test shows $p < 0.05$.

Value

A data frame of significant trait–site (or trait–protein) comparisons with:

- trait — glycan trait name
- level — site/protein identifier
- l_pval — Levene test p-value
- f_val — Levene test F statistic
- t_pval — Welch t-test p-value
- t_val — t-statistic

Rows correspond only to significant comparisons ($p < 0.05$) for ‘l_pval‘ or ‘t_pval‘.

Examples

```
# Load toy pGlyco3 GPSM data included with the package
path <- system.file("extdata", "pGlyco3_gpsm_toyexample.txt",
  package = "glycoTraitR"
)
gpsm_toyexample <- read_pGlyco3_gpsm(path)

# Load accompanying toy metadata
data("meta_toyexample")

# Build glycan trait SummarizedExperiment at the protein level
trait_se <- build_trait_se(
  gpsm = gpsm_toyexample,
  from = "pGlyco3",
  motifs = NULL,
  level = "protein",
  meta = meta_toyexample
)

# Identify glycan traits significantly changed between groups
changed_traits <- analyze_trait_changes(
  trait_se     = trait_se,
  group_col    = "Diagnosis",
  group_levels = c("Normal", "Symptomatic"),
  min_psm      = 20
)
changed_traits
```

`build_glycan_igraph` *Construct an igraph representation of a glycan tree*

Description

Convert a parsed glycan tree into a directed ‘igraph‘ object with parent–child relationships and residue-level metadata suitable for structural motif detection.

Usage

```
build_glycan_igraph(tree)
```

Arguments

`tree`	A parsed glycan tree from `pGlyco3_to_tree` or `wurcs_to_tree`.

Details

The resulting graph contains the following vertex attributes:

- `name` — synthetic node label (‘"a"‘, ‘"b"‘, ...)
- `residue` — residue type (H, N, A, F, G)
- `type` — identical to residue (for convenience)
- `color` — color encoding of residue type
- `is_root` — TRUE if the vertex is the structural root

Value

A directed ‘igraph‘ object representing the glycan structure.

Examples

```
# Example: parse a pGlyco3 monosaccharide expression into a glycan tree
pGlyco_expr <- "(N(N(H(H(H))(H(H)(H)(H(H))))))"

# Convert expression into a parsed tree structure
tree <- pGlyco3_to_tree(pGlyco_expr)
g <- build_glycan_igraph(tree)
g
```

`build_trait_se` *Build a SummarizedExperiment of glycan trait matrices*

Description

Convert a GPSM table into peptide- or protein-level glycan trait matrices and store them in a `SummarizedExperiment` object. Each trait becomes an assay matrix whose rows represent peptides or proteins, and whose columns represent individual GPSMs. This function provides a unified container for downstream analyses such as differential testing and visualization.

Usage

```
build_trait_se(gpsm, from, motifs = NULL, level, meta)
```

Arguments

`gpsm`	A GPSM table containing at least: ‘Protein‘, ‘Peptide‘, ‘GlycanStructure‘, ‘File‘, and ‘Count‘.
`from`	Character; glycan format used in the GPSM input. One of ‘decipher‘ or ‘pGlyco3‘.
`motifs`	Optional named list of user-defined motif structures passed to compute_glycan_traits.
`level`	Summarization level. Either ‘site (peptide)‘ or ‘"protein"‘.
`meta`	Data frame of sample metadata with a column ‘file‘ matching ‘gpsm$File‘.

Value

A ‘SummarizedExperiment‘ where each assay is a glycan-trait matrix (trait × PSM), ‘rowData‘ contains peptide/protein names, and ‘colData‘ contains metadata aligned to PSMs.

Examples

```
# Load toy GPSM table exported by pGlyco3
path <- system.file("extdata", "pGlyco3_gpsm_toyexample.txt",
  package = "glycoTraitR"
)
gpsm_toyexample <- read_pGlyco3_gpsm(path)

# Load toy metadata for summarization
data("meta_toyexample")


# Build glycan trait SummarizedExperiment at protein level
trait_se <- build_trait_se(
  gpsm = gpsm_toyexample,
  from = "pGlyco3",
  motifs = NULL,
  level = "protein",
  meta = meta_toyexample
)

# Inspect assay names and dimensions
SummarizedExperiment::assayNames(trait_se)
dim(trait_se)
```

`compute_glycan_traits` *Compute glycan traits from a parsed glycan tree*

Description

Combine residue-level composition traits (see `count_residues`), structural traits (see `compute_structural_traits`), and user-defined motifs (see `compute_userdefined_traits`) into a unified trait vector.

Usage

```
compute_glycan_traits(tree, motifs)
```

Arguments

`tree`	A parsed glycan tree from `pGlyco3_to_tree` or `wurcs_to_tree`.
`motifs`	Optional named list of user-defined glycan motifs.

Value

A named list of numeric trait values.

Examples

```
# Example: parse a pGlyco3-style glycan expression into a tree
pGlyco_expr <- "(N(N(H(H(H))(H(H)(H)(H(H))))))"

# Convert to glycan tree structure
tree <- pGlyco3_to_tree(pGlyco_expr)

# Explore parsed nodes and edges
tree$node
tree$edge

# Build igraph representation
g <- build_glycan_igraph(tree)
plot_glycan_tree(g)

# Define user motifs for trait computation
user_motifs <- list(
  LinearH3 = list(
    node = c("H", "H", "H"),
    edge = c("a-b", "b-c")
  ),
  FucBranch = list(
    node = c("H", "N", "F"),
    edge = c("a-b", "b-c")
  )
)

# Compute glycan structural traits
compute_glycan_traits(tree, motifs = user_motifs)
```

`glycanDatabase` *Glycan annotation reference database*

Description

A curated reference table mapping glycan IDs to their structures, used internally by glycoTraitR and also available to users.

Usage

```
data(glycanDatabase)
```

Format

A data frame with N rows and M columns.

Examples

```
data(glycanDatabase)
head(glycanDatabase)
```

`meta_toyexample` *Toy metadata for glycoTraitR examples*

Description

A toy example metadata table used in vignettes, examples, and tests.

Usage

```
data(meta_toyexample)
```

Format

A data frame with 34 rows and 2 variables:

Diagnosis Clinical diagnosis (factor)

Sample number Sample ID (numeric)

file File Name (character)

Examples

```
data(meta_toyexample)
head(meta_toyexample)
```

`pGlyco3_to_tree` *Convert a pGlyco3 glycan string into a glycan tree structure*

Description

Parse a pGlyco3-style glycan expression (e.g. `"N(H(H))"`) and reconstruct the residue sequence and edge relationships as a tree suitable for downstream structural analysis. This parser assumes simple pGlyco3 monosaccharide symbols (e.g. "N", "H", "A", "F").

Usage

```
pGlyco3_to_tree(expr)
```

Arguments

`expr`	A character string representing the pGlyco3 glycan structure.

Details

This function interprets parentheses as branch delimiters and assigns:

1. one residue per character (e.g. `N`, `H`, `A`)
2. parent–child edges based on bracket nesting

Each residue is assigned a synthetic node label (`a`, `b`, `c`, ...), ensuring compatibility with graph-based trait extraction.

Value

A list with:

- `node`: character vector of residue types
- `edge`: character vector of edges in "a-b" format

Examples

```
# Example: parse a pGlyco3-style glycan expression into a tree
pGlyco_expr <- "(N(N(H(H(H))(H(H)(H)(H(H))))))"
# Convert to glycan tree structure
tree <- pGlyco3_to_tree(pGlyco_expr)
tree
```

`plot_glycan_tree` *Plot a glycan structure represented as an igraph tree*

Description

Visualize a glycan structure encoded as an igraph object produced by `build_glycan_igraph`. This plot is mainly intended for inspecting glycan topology (branching, residue types, connectivity).

Usage

```
plot_glycan_tree(g)
```

Arguments

`g`	An igraph object representing a glycan tree from `build_glycan_igraph`.

Value

A glycan topology plot is drawn as a side effect.

Examples

```
# Example: parse a pGlyco3-style glycan expression into a tree
pGlyco_expr <- "(N(N(H(H(H))(H(H)(H)(H(H))))))"

# Convert to glycan tree structure
tree <- pGlyco3_to_tree(pGlyco_expr)

# Explore parsed nodes and edges
tree$node
tree$edge

# Build igraph representation
g <- build_glycan_igraph(tree)
plot_glycan_tree(g)
```

`plot_trait_distribution` *Plot the distribution of a glycan trait across experimental groups*

Description

Generate two diagnostic plots—a histogram and a boxplot—for a selected glycan trait at a specific site/protein feature. This function extracts GPSM-level values from a trait matrix stored in a ‘SummarizedExperiment‘, subsets them by group, removes missing values, and visualizes the resulting distribution.

Usage

```
plot_trait_distribution(trait_se, group_col, group_levels, trait_name, feature)
```

Arguments

`trait_se`	A SummarizedExperiment object generated by build_trait_se, containing one assay matrix per glycan trait.
`group_col`	Column name in ‘colData(trait_se)‘ that defines sample group membership.
`group_levels`	Character vector specifying the group values to include in the plot. Typically two values (e.g. ‘c("Control","Treatment")‘).
`trait_name`	Name of the glycan trait to plot. Must match an assay name in ‘assays(trait_se)‘.
`feature`	Row identifier within the trait matrix (peptide, or protein). Must match a row name in ‘assays(trait_se)[[trait_name]]‘.

Value

A named list of two ‘ggplot2‘ objects: * ‘p_hist‘ — histogram of trait intensities * ‘p_box‘ — boxplot with jitter overlay

Examples

```
# Load the toy GPSM table exported by pGlyco3 (included in the package)
path <- system.file("extdata", "pGlyco3_gpsm_toyexample.txt",
  package = "glycoTraitR"
)
gpsm_toyexample <- read_pGlyco3_gpsm(path)

# Load the accompanying toy metadata
data("meta_toyexample")

# Build a protein-level glycan trait SummarizedExperiment object
trait_se <- build_trait_se(
  gpsm = gpsm_toyexample,
  from = "pGlyco3",
  motifs = NULL,
  level = "protein",
  meta = meta_toyexample
)

# Identify glycan traits significantly changed between two groups
changed_traits <- analyze_trait_changes(
  trait_se     = trait_se,
  group_col    = "Diagnosis",
  group_levels = c("Normal", "Symptomatic"),
  min_psm      = 20
)

# Extract one trait name and one protein/site feature to visualize
trait_name <- changed_traits$trait[1]
feature <- changed_traits$level[1]
```

```
# Plot the distribution of this selected trait
p <- plot_trait_distribution(
  trait_se     = trait_se,
  group_col    = "Diagnosis",
  group_levels = c("Normal", "Symptomatic"),
  trait_name   = trait_name,
  feature      = feature
)

# Show histogram and boxplot
p$p_hist
p$p_box
```

`read_decipher_gpsm` *Combine Glyco-Decipher GPSM results into a long-format table*

Description

Read multiple Glyco-Decipher GPSM files from a folder, merge them into a unified protein–peptide–glycan table, and attach glycan structures (WURCS 2.0).

Usage

```
read_decipher_gpsm(gpsm_folder_dir)
```

Arguments

`gpsm_folder_dir`	The path to a folder containing Glyco-Decipher GPSM files (e.g. files ending with `"_GPSM_DatabaseGlycan.txt"`).

Details

This function assumes that the input folder contains one or more Glyco-Decipher GPSM files, typically named with the suffix `"_GPSM_DatabaseGlycan.txt"`. For each file, GPSM records are read and collapsed by `Protein`, `Peptide`, `GlycanID`, `File`, and `Count`. All files are then combined into a single table, and glycan IDs are mapped to WURCS structures via the global `glycanDatabase` object. The final table uses a standardized glycan column name `GlycanStructure` for compatibility with downstream functions.

Value

A data frame in long format with one row per `Protein–Peptide–GlycanStructure–File` combination and the following columns:

- `Protein` — protein identifier(s)
- `Peptide` — peptide sequence

- `GlycanStructure` — glycan structural annotation (WURCS 2.0)
- `File` — raw file name
- `Count` — spectral count (number of GPSMs) for this combination

The returned table is designed to be passed to `build_trait_se` for glycan trait computation.

See Also

`read_pGlyco3_gpsm`, `build_trait_se`

Examples

```
folder <- system.file("extdata", "decipher_toyexample", package = "glycoTraitR")
gpsm <- read_decipher_gpsm(folder)
head(gpsm)
```

`read_pGlyco3_gpsm` *Import pGlyco3 GPSM results as a long-format table*

Description

Read a pGlyco3 GPSM result file and convert it into a long-format protein–peptide–glycan table with spectral counts per raw file. Each row corresponds to a unique combination of protein, peptide, glycan structure, and file.

Usage

```
read_pGlyco3_gpsm(gpsm_dir)
```

Arguments

`gpsm_dir`	The path to the pGlyco3 GPSM output file (for example, `"pGlycoDB-GP-FDR-Pro-Quant-Site.txt"`).

Details

This function takes a pGlyco3 GPSM file as input (typically named `pGlycoDB-GP-FDR-Pro-Quant-Site.txt`). The following steps are performed:

- Select relevant columns (`RawName`, `Proteins`, `Peptide`, `PlausibleStruct`).
- Rename them to a standardized schema: `File`, `Protein`, `Peptide`, `GlycanStructure`.
- Collapse multiple protein IDs per PSM into a single pipe-separated string (e.g. `"P00123|P00456"`).
- Aggregate rows by `Protein`, `Peptide`, `GlycanStructure`, and `File` as GPSM counts in each group.

The output of this function is typically used as the input for `build_trait_se`.

Value

A data frame with one row per `Protein`–`Peptide`–`GlycanStructure`–`File` combination and the following columns:

- `Protein` — protein identifier(s)
- `Peptide` — peptide sequence
- `GlycanStructure` — glycan structural annotation from pGlyco3
- `File` — raw file name
- `Count` — spectral count (number of GPSMs) for this combination

See Also

`read_decipher_gpsm`, `build_trait_se`

Examples

```
# Load toy example data included in glycoTraitR
path <- system.file("extdata", "pGlyco3_gpsm_toyexample.txt",
  package = "glycoTraitR"
)
gpsm <- read_pGlyco3_gpsm(path)
head(gpsm)
```

`wurcs_to_tree` *Convert a WURCS string into a glycan tree structure*

Description

Parse a WURCS (WURCS 2.0) glycan annotation and extract:

- residue sequence (as a character vector)
- edge list (parent–child relationships)

Usage

```
wurcs_to_tree(w)
```

Arguments

`w`	A character string containing a WURCS 2.0 glycan annotation.

Details

This function is used internally to convert WURCS strings into a tree representation that can support structural trait computation and graph construction.

The function performs several parsing steps:

1. Remove the WURCS prefix and split the string into components.
2. Extract *UniqueRES* entries and map them to residue symbols using the internal `WURCS_RES_MAP` table.
3. Follow the *RES sequence* index to reconstruct the residue vector.
4. Parse *LIN* entries and normalize them into simple "X-Y" edges.

Value

A list with two elements:

- `node`: character vector of residue types in order
- `edge`: character vector of edges in "A-B" format

Examples

```
# Example WURCS glycan string
w <- paste0(
  "WURCS=2.0/4,9,8/",
  "[u2122h_2*NCC/3=O]",
  "[a2122h-1b_1-5_2*NCC/3=O]",
  "[a1122h-1b_1-5]",
  "[a1122h-1a_1-5]",
  "/1-2-3-4-4-4-4-4-4/",
  "a4-b1_b4-c1_c3-d1_c6-f1_d2-e1_f3-g1_f6-h1_h2-i1"
)
tree <- wurcs_to_tree(w)
tree
```

Index