Package: glyenzy (via r-universe)

June 8, 2026

Title Deconstruction Glycosylation Biosynthesis

Version 0.5.2

Description Simulates the biosynthetic process of glycosylation in silico, enabling computational analysis of glycan biosynthesis pathways. Key features include: identifying specific glycosyltransferases responsible for each glycosidic bond at the bond level; reconstructing complete biosynthetic pathways for given glycans; exploring possible transformations between two glycan structures via glycosyltransferases and glycosidases; and predicting glycans that can be synthesized from substrate structures and enzyme sets. The package supports multiple glycan types including N-glycans, O-GalNAc, O-Man, O-GlcNAc, O-Fuc, and O-Glc. As a downstream component of the glycoverse ecosystem, it integrates seamlessly with 'glyrepr', 'glyparse', and 'glymotif' to provide practical, analysis-ready insights into glycan biosynthesis for omics researchers.

License MIT + file LICENSE

Suggests ggplot2, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

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

Imports checkmate, cli, glydraw (>= 0.4.0), glymotif (>= 0.12.1), glyrepr (>= 0.10.0), glyparse (>= 0.5.3), igraph, purrr, fastmap, dplyr, rlang, tibble, stringr, R6

Depends R (>= 4.1)

LazyData true

VignetteBuilder knitr

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

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

Date/Publication 2026-06-08 05:22:15 UTC

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

RemoteRef v0.5.2

RemoteSha decfd598c3bcdd88e8a7f59dc43d347599b00397

Contents

`apply_enzyme` *Apply an Enzyme to a Glycan*

Description

This function simulates the action of an enzyme on a glycan. It returns all possible products generated by the enzyme with the given glycans.

Usage

```
apply_enzyme(glycans, enzyme, return_list = NULL)
```

Arguments

`glycans`	A `glyrepr::glycan_structure()`, or a character vector of glycan structure strings supported by `glyparse::auto_parse()`.
`enzyme`	An `enzyme()` or a gene symbol.
`return_list`	If `NULL` (default), return a list of `glyrepr::glycan_structure()` when `glycans` has length greater than 1, and a single `glyrepr::glycan_structure()` when `glycans` has length 1. Set to `TRUE` to always return a list. This can be useful when you are working programmatically with unknown input length. Note that when `return_list = FALSE` and `length(glycans) > 1`, an error will be thrown.

Value

A `glyrepr::glycan_structure()` vector, or a list of such vectors.

Important notes

Here are some important notes for all functions in the `glyenzy` package.

Applicability:

All algorithms and enzyme information in glyenzy are applicable only to humans, and specifically to N-glycans and O-GalNAc glycans. Results may be inaccurate for other types of glycans (e.g., GAGs, glycolipids) or for glycans in other species (e.g., plants, insects).

Inclusiveness:

The algorithm takes an intentionally inclusive approach, assuming that all possible isoenzymes capable of catalyzing a given reaction may be involved. Therefore, results should be interpreted with caution.

For example, in humans, detection of the motif "Neu5Ac(a2-3)Gal(b1-" will return both "ST3GAL3" and "ST3GAL4". In reality, only one of them might be active, depending on factors such as tissue specificity.

Only "concrete" glycans:

The function only works for glycans containing **concrete** residues (e.g., `"Glc"`, `"GalNAc"`), and not for glycans with **generic** residues (e.g., `"Hex"`, `"HexNAc"`).

Substituents:

Substituents (e.g. sulfation, phosphorylation) are not supported yet, and the algorithms might fail for glycans with substituents. If your glycans contain substituents, use `glyrepr::remove_substituents()` to get clean glycans.

Incomplete glycan structures:

If the glycan structure is incomplete or partially degraded, the result may be misleading.

Starting points:

- For N-glycans, the starting structure is assumed to be "Glc(3)Man(9)GlcNAc(2)", the N-glycan precursor transferred to Asn by OST.
- For O-GalNAc glycans, the starting structure is assumed to be "GalNAc(a1-".
- For O-GlcNAc glycans, the starting structure is assumed to be "GlcNAc(b1-".
- For O-Man glycans, the starting structure is assumed to be "Man(a1-".
- For O-Fuc glycans, the starting structure is assumed to be "Fuc(a1-".
- For O-Glc glycans, the starting structure is assumed to be "Glc(b1-".

Examples

```
library(glyrepr)
library(glyparse)

# Use `glycan_structure()` and `enzyme()`
glycan <- auto_parse("GlcNAc(b1-2)Man(a1-3)[Man(a1-6)]Man(b1-4)GlcNAc(b1-4)GlcNAc(b1-")
```

```
apply_enzyme(glycan, enzyme("MGAT3"))

# Or use characters directly
apply_enzyme("GlcNAc(b1-2)Man(a1-3)[Man(a1-6)]Man(b1-4)GlcNAc(b1-4)GlcNAc(b1-", "MGAT3")

# Vectorized input
glycans <- c(
  "GlcNAc(b1-2)Man(a1-3)[Man(a1-6)]Man(b1-4)GlcNAc(b1-4)GlcNAc(b1-",
  "GlcNAc(b1-2)Man(a1-3)[GlcNAc(b1-2)Man(a1-6)]Man(b1-4)GlcNAc(b1-4)GlcNAc(b1-"
)
apply_enzyme(glycans, "MGAT3")
```

`count_enzyme` *Count Enzyme Involvement*

Description

Count how many times an enzyme is involved in the biosynthesis of a glycan.

Usage

```
count_enzyme(glycans, enzyme)
```

Arguments

`glycans`	A `glyrepr::glycan_structure()`, or a character vector of glycan structure strings supported by `glyparse::auto_parse()`.
`enzyme`	An `enzyme()` or a gene symbol.

Value

An integer vector of the same length as `glycans`.

Important notes

Here are some important notes for all functions in the `glyenzy` package.

Applicability:
All algorithms and enzyme information in glyenzy are applicable only to humans, and specifically to N-glycans and O-GalNAc glycans. Results may be inaccurate for other types of glycans (e.g., GAGs, glycolipids) or for glycans in other species (e.g., plants, insects).

Inclusiveness:
The algorithm takes an intentionally inclusive approach, assuming that all possible isoenzymes capable of catalyzing a given reaction may be involved. Therefore, results should be interpreted with caution.
For example, in humans, detection of the motif "Neu5Ac(a2-3)Gal(b1-" will return both "ST3GAL3" and "ST3GAL4". In reality, only one of them might be active, depending on factors such as tissue specificity.

Only "concrete" glycans:

The function only works for glycans containing **concrete** residues (e.g., `"Glc"`, `"GalNAc"`), and not for glycans with **generic** residues (e.g., `"Hex"`, `"HexNAc"`).

Substituents:

Substituents (e.g. sulfation, phosphorylation) are not supported yet, and the algorithms might fail for glycans with substituents. If your glycans contain substituents, use `glyrepr::remove_substituents()` to get clean glycans.

Incomplete glycan structures:

If the glycan structure is incomplete or partially degraded, the result may be misleading.

Starting points:

- For N-glycans, the starting structure is assumed to be "Glc(3)Man(9)GlcNAc(2)", the N-glycan precursor transferred to Asn by OST.
- For O-GalNAc glycans, the starting structure is assumed to be "GalNAc(a1-".
- For O-GlcNAc glycans, the starting structure is assumed to be "GlcNAc(b1-".
- For O-Man glycans, the starting structure is assumed to be "Man(a1-".
- For O-Fuc glycans, the starting structure is assumed to be "Fuc(a1-".
- For O-Glc glycans, the starting structure is assumed to be "Glc(b1-".

Examples

```
library(glyrepr)
library(glyparse)

# Use `glycan_structure()` and `enzyme()`
glycan <- auto_parse("Neu5Ac(a2-6)Gal(b1-4)GlcNAc(b1-")
count_enzyme(glycan, enzyme("ST6GAL1"))

# Or use characters directly
count_enzyme("Neu5Ac(a2-6)Gal(b1-4)GlcNAc(b1-", "ST6GAL1")

# Vectorized input
glycans <- c(
  "Neu5Ac(a2-6)Gal(b1-4)GlcNAc(b1-",
  "Gal(b1-4)GlcNAc(b1-"
)
count_enzyme(glycans, "ST6GAL1")
```

`db_enzymes` *Get all enzymes*

Description

Return a named list of all built-in enzymes, or a character vector of gene symbols if `return_str` is `TRUE`.

Usage

```
db_enzymes(
  return_str = FALSE,
  include_starter_gt = TRUE,
  include_npre_gt = TRUE
)
```

Arguments

return_str	If FALSE (default), returns the enzyme list. Otherwise returns a character vector of gene symbols.
include_starter_gt	If TRUE (default), includes starter GTs in the result. Starter GTs are enzymes that initiate glycosylation by introducing the first sugar residue onto a non-glycan substrate. For example, DPAGT1 is the starter GT for N-glycosylation.
include_npre_gt	If TRUE (default), includes GTs involved in N-glycan precursor synthesis in the result. These GTs are responsible for building the N-glycan precursor before it is transferred to the target protein by OST.

Value

A list of enzyme()s or a character vector.

Examples

```
db_enzymes(return_str = TRUE)
```

---

enzyme *Enzymes*

Description

Two types of enzymes are involved in glycosylation: glycosyltransferases (GTs) and glycoside hydrolases (GHs).

- GTs catalyze the transfer of a sugar residue from a donor to an acceptor, thus building up glycan structures.
- GHs catalyze the removal of a sugar residue from a substrate, thus breaking down glycan structures.

One special subtype of GTs are starter GTs (or initiating GTs), which catalyze the addition of the first sugar residue onto a non-glycan substrate, thus initiating glycosylation.

Use enzyme() with a gene symbol to load a predefined enzyme. For example, use enzyme("ST3GAL3") to load the enzyme ST3GAL3.

Throughout the package, you can use enzyme()s for any enzyme argument, or just use the gene symbol directly. For example, involve("Neu5Ac(a2-3)Gal(b1-3)GalNAc(a1-", "ST3GAL3") and involve("Neu5Ac(a2-3)Gal(b1-3)GalNAc(a1-", enzyme("ST3GAL3")) are equivalent.

Usage

```
enzyme(symbol)
```

Arguments

symbol	The gene symbol of the enzyme.

Value

A glyenzy_enzyme object.

Explanation about glyenzy_enzyme

An enzyme() is a list with the following elements:

1. name: the name of the enzyme, usually the gene symbol.
2. rules: a list of glyenzy_enzyme_rule objects. Each rule is a list with the following fields:
   - acceptor: the motif that the enzyme recognizes
   - acceptor_alignment: the alignment of the acceptor
   - rejects: the motifs that the enzyme rejects to act on
   - product: the product generated by the enzyme
   - acceptor_idx: the node index of the acceptor where the enzyme acts on. For GTs, this is the node new residue is attached to. For GHs, this is the node that is removed. For starter GTs, acceptor_idx is 0.
   - product_idx: the node index of the product residue in the product structure. For GTs, this is the index of the newly added residue in the product. For GHs, this is NULL (no new residue is added).
   - new_residue: the new residue added by the enzyme. For GHs, this is NULL.
   - new_linkage: the linkage of the new residue. For GHs, this is NULL.
3. type: the broad type of the enzyme, "GT" for glycosyltransferase or "GH" for glycoside hydrolase. Starter GTs are encoded as type = "GT".
4. species: the species of the enzyme, e.g. "human" or "mouse".

You can see all these information by printing the enzyme object.

Examples

```
library(glyrepr)

enzyme("ST3GAL3")
```

`find_enzyme` *Identify Potentially Involved Enzymes*

Description

This function returns all possible isoenzymes associated with the biosynthetic steps of the input glycan. Note that this function ignores the residues in glycans that cannot be matched to any enzyme rules.

Usage

```
find_enzyme(glycans, return_list = NULL)
```

Arguments

`glycans`	A `glyrepr::glycan_structure()`, or a character vector of glycan structure strings supported by `glyparse::auto_parse()`.
`return_list`	If `NULL` (default), return a list of character vectors when `glycans` has length greater than 1, and a single character vector when `glycans` has length 1. Set to `TRUE` to always return a list. This can be useful when you are working programmatically with unknown input length. Note that when `return_list = FALSE` and `length(glycans) > 1`, an error will be thrown.

Value

A character vector or a list of character vectors (see `return_list` parameter), each containing the names of enzymes involved in the biosynthesis of the corresponding glycan.

Important notes

Here are some important notes for all functions in the `glyenzy` package.

Applicability:

All algorithms and enzyme information in glyenzy are applicable only to humans, and specifically to N-glycans and O-GalNAc glycans. Results may be inaccurate for other types of glycans (e.g., GAGs, glycolipids) or for glycans in other species (e.g., plants, insects).

Inclusiveness:

The algorithm takes an intentionally inclusive approach, assuming that all possible isoenzymes capable of catalyzing a given reaction may be involved. Therefore, results should be interpreted with caution.

For example, in humans, detection of the motif "Neu5Ac(a2-3)Gal(b1-" will return both "ST3GAL3" and "ST3GAL4". In reality, only one of them might be active, depending on factors such as tissue specificity.

Only "concrete" glycans:

The function only works for glycans containing **concrete** residues (e.g., `"Glc"`, `"GalNAc"`), and not for glycans with **generic** residues (e.g., `"Hex"`, `"HexNAc"`).

Substituents:

Substituents (e.g. sulfation, phosphorylation) are not supported yet, and the algorithms might fail for glycans with substituents. If your glycans contain substituents, use `glyrepr::remove_substituents()` to get clean glycans.

Incomplete glycan structures:

If the glycan structure is incomplete or partially degraded, the result may be misleading.

Starting points:

- For N-glycans, the starting structure is assumed to be "Glc(3)Man(9)GlcNAc(2)", the N-glycan precursor transferred to Asn by OST.
- For O-GalNAc glycans, the starting structure is assumed to be "GalNAc(a1-".
- For O-GlcNAc glycans, the starting structure is assumed to be "GlcNAc(b1-".
- For O-Man glycans, the starting structure is assumed to be "Man(a1-".
- For O-Fuc glycans, the starting structure is assumed to be "Fuc(a1-".
- For O-Glc glycans, the starting structure is assumed to be "Glc(b1-".

Examples

```
library(glyrepr)
library(glyparse)

# Use `glycan_structure()`
glycans <- auto_parse(c(
  "GlcNAc(b1-2)Man(a1-3)[Man(a1-6)]Man(b1-4)GlcNAc(b1-4)GlcNAc(b1-",
  "GlcNAc(b1-2)Man(a1-3)[GlcNAc(b1-2)Man(a1-6)]Man(b1-4)GlcNAc(b1-4)GlcNAc(b1-"
))
find_enzyme(glycans)

# Or use characters directly
find_enzyme("GlcNAc(b1-2)Man(a1-3)[Man(a1-6)]Man(b1-4)GlcNAc(b1-4)GlcNAc(b1-")
```

`grow_glycans_step` *Grow Glycans with Enzymes*

Description

This function simulates the action of enzymes on glycans. Think of it like a primordial soup where you put in a few glycans and enzymes, and let them interact to generate new glycans.

`grow_glycans_step()` performs one round of enzyme action, while `grow_glycans()` performs multiple rounds. The only difference between `grow_glycans_step()` and `grow_glycans(n_steps = 1)` is that the latter returns the original input glycans as well. For both, a vector of unique glycan structures is returned.

The number of glycans generated by `grow_glycans()` is typically exponential, and can quickly become very large. Therefore, it is recommended to use a small number of steps and carefully select the enzymes. Also, you can use the `filter` argument to prune the results after each round.

Usage

```
grow_glycans_step(glycans, enzymes)

grow_glycans(glycans, enzymes, n_steps = 5, filter = NULL)
```

Arguments

glycans	A `glyrepr::glycan_structure()`, or a character vector of glycan structure strings supported by `glyparse::auto_parse()`.
enzymes	A character vector of gene symbols, or a list of `enzyme()` objects.
n_steps	The maximum number of rounds to perform. The actual number of rounds may be less if no new glycans can be generated.
filter	A function to filter the generated glycans. It should have a single `glyrepr::glycan_structure()` vector as input, and return a logical vector of the same length. The function will be called on the results after each round.

Value

A `glyrepr::glycan_structure()` vector of all unique glycans generated.

Important notes

Here are some important notes for all functions in the `glyenzy` package.

Applicability:

All algorithms and enzyme information in glyenzy are applicable only to humans, and specifically to N-glycans and O-GalNAc glycans. Results may be inaccurate for other types of glycans (e.g., GAGs, glycolipids) or for glycans in other species (e.g., plants, insects).

Inclusiveness:

The algorithm takes an intentionally inclusive approach, assuming that all possible isoenzymes capable of catalyzing a given reaction may be involved. Therefore, results should be interpreted with caution.

For example, in humans, detection of the motif "Neu5Ac(a2-3)Gal(b1-" will return both "ST3GAL3" and "ST3GAL4". In reality, only one of them might be active, depending on factors such as tissue specificity.

Only "concrete" glycans:

The function only works for glycans containing **concrete** residues (e.g., `"Glc"`, `"GalNAc"`), and not for glycans with **generic** residues (e.g., `"Hex"`, `"HexNAc"`).

Substituents:

Substituents (e.g. sulfation, phosphorylation) are not supported yet, and the algorithms might fail for glycans with substituents. If your glycans contain substituents, use `glyrepr::remove_substituents()` to get clean glycans.

Incomplete glycan structures:

If the glycan structure is incomplete or partially degraded, the result may be misleading.

Starting points:

- For N-glycans, the starting structure is assumed to be "Glc(3)Man(9)GlcNAc(2)", the N-glycan precursor transferred to Asn by OST.
- For O-GalNAc glycans, the starting structure is assumed to be "GalNAc(a1-".
- For O-GlcNAc glycans, the starting structure is assumed to be "GlcNAc(b1-".
- For O-Man glycans, the starting structure is assumed to be "Man(a1-".
- For O-Fuc glycans, the starting structure is assumed to be "Fuc(a1-".
- For O-Glc glycans, the starting structure is assumed to be "Glc(b1-".

Examples

```
# Use `grow_glycans_step()` to build glycans step by step
glycan <- "GlcNAc(b1-2)Man(a1-3)[Man(a1-6)]Man(b1-4)GlcNAc(b1-4)GlcNAc(b1-"
glycan |>
  grow_glycans_step("MGAT2") |>
  grow_glycans_step("B4GALT1") |>
  grow_glycans_step("ST3GAL3")

# Use `grow_glycans()` to simulate a primordial soup
glycans <- c(
  "GlcNAc(b1-2)Man(a1-3)[Man(a1-6)]Man(b1-4)GlcNAc(b1-4)GlcNAc(b1-",
  "GlcNAc(b1-2)Man(a1-3)[GlcNAc(b1-2)Man(a1-6)]Man(b1-4)GlcNAc(b1-4)GlcNAc(b1-"
)
enzymes <- c("B4GALT1", "ST3GAL3")
grow_glycans(glycans, enzymes, n_steps = 5)

# Use `filter` to prune the results after each round
# Here we keep only glycans that are synthesized by MGAT2
grow_glycans(glycans, enzymes, n_steps = 5, filter = ~ have_enzyme(.x, "MGAT2"))
```

---

have_enzyme *Determine Whether a Glycan Is Synthesized by a Given Enzyme*

Description

Glycans are produced through a series of enzymatic reactions. This function checks whether a specific enzyme participates in the biosynthesis of a given glycan (or glycans).

Usage

```
have_enzyme(glycans, enzyme)
```

Arguments

glycans	A `glyrepr::glycan_structure()`, or a character vector of glycan structure strings supported by `glyparse::auto_parse()`.
enzyme	An `enzyme()` or a gene symbol.

Value

A logical vector of the same length as `glycans`.

Important notes

Here are some important notes for all functions in the `glyenzy` package.

Applicability:
All algorithms and enzyme information in glyenzy are applicable only to humans, and specifically to N-glycans and O-GalNAc glycans. Results may be inaccurate for other types of glycans (e.g., GAGs, glycolipids) or for glycans in other species (e.g., plants, insects).

Inclusiveness:
The algorithm takes an intentionally inclusive approach, assuming that all possible isoenzymes capable of catalyzing a given reaction may be involved. Therefore, results should be interpreted with caution.
For example, in humans, detection of the motif "Neu5Ac(a2-3)Gal(b1-" will return both "ST3GAL3" and "ST3GAL4". In reality, only one of them might be active, depending on factors such as tissue specificity.

Only "concrete" glycans:
The function only works for glycans containing **concrete** residues (e.g., `"Glc"`, `"GalNAc"`), and not for glycans with **generic** residues (e.g., `"Hex"`, `"HexNAc"`).

Substituents:
Substituents (e.g. sulfation, phosphorylation) are not supported yet, and the algorithms might fail for glycans with substituents. If your glycans contain substituents, use `glyrepr::remove_substituents()` to get clean glycans.

Incomplete glycan structures:
If the glycan structure is incomplete or partially degraded, the result may be misleading.

Starting points:

- For N-glycans, the starting structure is assumed to be "Glc(3)Man(9)GlcNAc(2)", the N-glycan precursor transferred to Asn by OST.
- For O-GalNAc glycans, the starting structure is assumed to be "GalNAc(a1-".
- For O-GlcNAc glycans, the starting structure is assumed to be "GlcNAc(b1-".
- For O-Man glycans, the starting structure is assumed to be "Man(a1-".
- For O-Fuc glycans, the starting structure is assumed to be "Fuc(a1-".
- For O-Glc glycans, the starting structure is assumed to be "Glc(b1-".

Algorithm

The basic approach is straightforward: for each reaction rule associated with the enzyme, the function checks whether the corresponding product motif appears in the glycan. If any rule matches, the function returns `TRUE`.

For N-glycans, additional logic is applied to handle special cases. Products of **MGAT1** are often further trimmed by glycoside hydrolases, meaning that the final glycan product may no longer contain the original motif. In these cases, the function instead looks for specific motif markers to determine enzyme involvement.

Examples

```
library(glyrepr)
library(glyparse)

# Use `glycan_structure()` and `enzyme()`
glycan <- auto_parse("Neu5Ac(a2-6)Gal(b1-4)GlcNAc(b1-")
have_enzyme(glycan, enzyme("ST6GAL1"))

# Or use characters directly
have_enzyme("Neu5Ac(a2-6)Gal(b1-4)GlcNAc(b1-", "ST6GAL1")

# Vectorized input
glycans <- c(
  "Neu5Ac(a2-6)Gal(b1-4)GlcNAc(b1-",
  "Gal(b1-4)GlcNAc(b1-"
)
have_enzyme(glycans, "ST6GAL1")
```

---

make_enzyme *Make a Custom Enzyme*

Description

This function creates a custom enzyme object, which can be used in all functions in this package that accept an enzyme argument. Most of the time, you can use `enzyme()` with a gene symbol to get a predefined enzyme, or just use the gene symbol directly in function arguments. Use this function only when you need to create an enzyme with custom rules. See the "Enzyme rules" section below for more details.

For now you can only create regular glycosyltransferases (GTs) and glycoside hydrolases (GHs) with this function. Special enzymes like starter GTs for glycosylation initiation are reserved for predefined enzymes.

Usage

```
make_enzyme(name, rules, type, species)
```

Arguments

`name`	The name of the enzyme.
`rules`	A list of enzyme rules, each rule being a list with the following fields: • `acceptor`: A `glyrepr::glycan_structure()` scalar or any structure string supported by `glyparse::auto_parse()`, the motif recognized by the enzyme. • `acceptor_alignment`: A character string, the alignment of the `acceptor`. Possible values are "substructure", "core", "terminal" and "whole". See `glymotif::have_motif()` for details.

- rejects: A glyrepr::glycan_structure() vector or any structure character vector supported by glyparse::auto_parse(), the motifs that the enzyme rejects. Can also be NULL.
- product: A glyrepr::glycan_structure() scalar or any structure string supported by glyparse::auto_parse(), the product generated by the enzyme.

type — The type of the enzyme, "GT" for glycosyltransferase or "GH" for glycoside hydrolase.

species — The species of the enzyme. Default is "human". Currently no validation is done on the species. This field is only used for information purposes.

Value

A glyenzy_enzyme object.

Enzyme rules

The enzyme rules are the core of an enzyme object. Each rule defines one possible reaction of an enzyme, and an enzyme can have multiple rules.

We now explain each field of an enzyme rule in detail.

acceptor:

The acceptor is the glycan motif (substructure) that the enzyme recognizes. It can be as simple as a single residue, like Man(a1-, or a more complex motif, like Man(a1-3)[Man(a1-6)]Man(b1-4)GlcNAc(b1-4)GlcNAc(b1
Note that the acceptor does not define where the enzyme acts on, which is determined collectively by the acceptor and product fields.

acceptor_alignment:

Under the hood, the acceptor is detected by glymotif::match_motif(). To match a motif, we need to specify the alignment of the motif. Possible values are "substructure", "core", "terminal" and "whole". See glymotif::have_motif() for details.

rejects:

Some enzymes are really picky. They will reject to act on certain glycans if some motifs are present, even if the acceptor is perfectly matched. For example, if we want to define an enzyme that adds a Neu5Ac residue to the Gal residue of a Gal(b1-4)GlcNAc(b1- motif, but only when there is no a1-3 Fuc on that GlcNAc, we can set "Gal(b1-4)GlcNAc(b1-" as the acceptor and "Fuc(a1-3)[Gal(b1-4)]GlcNAc(b1-" as the rejects. You can provide multiple rejects as a vector. For example, if we want to reject Gal(b1-4)GlcNAc(b1- with either a1-3 Fuc on GlcNAc or a1-2 Fuc on Gal, we can set c("Fuc(a1-3)[Gal(b1-4)]GlcNAc(b1-", "Fuc(a1-2)Gal(b1-4)GlcNAc(b1-") as the rejects.

Note that here are some restrictions on the rejects:

- When acceptor_alignment is "whole", rejects cannot be set.
- The acceptor must be the substructure of all rejects. For example, in the a1-3 Fuc example above, you cannot just set "Fuc(a1-3)GlcNAc(b1-" as the rejects, because "Gal(b1-4)GlcNAc(b1-" (the acceptor) is not the substructure of "Fuc(a1-3)GlcNAc(b1-" (the reject).
- rejects cannot be the same structure as acceptor.

If `rejects` is `NULL`, it means the enzyme does not reject any motifs.

product:

The `product` is what the `acceptor` is transformed into by the enzyme. For glycosyltransferases (GTs), the `product` should have exactly one more residue than the `acceptor`. For glycoside hydrolases (GHs), the `product` should have exactly one less residue than the `acceptor`. If you have multiple possible products, put them in different rules. Take the same example above, if we want to add a Neu5Ac residue to the Gal residue of a `Gal(b1-4)GlcNAc(b1-` motif, we can set `"Neu5Ac(a2-3)Gal(b1-4)GlcNAc(b1-"` as the `product`.

See Also

`enzyme()`

Examples

```
library(glyrepr)

# Create a custom enzyme that adds a Neu5Ac residue to the Gal residue
make_enzyme(
  name = "MySiaT",
  rules = list(
    list(
      acceptor = "Gal(b1-4)GlcNAc(b1-",
      acceptor_alignment = "terminal",
      rejects = NULL,
      product = "Neu5Ac(a2-3)Gal(b1-4)GlcNAc(b1-"
    )
  ),
  type = "GT",
  species = "human"
)

# Create a custom enzyme that removes a Gal residue from a Gal(b1-4)GlcNAc(b1-` motif
make_enzyme(
  name = "MyGalH",
  rules = list(
    list(
      acceptor = "Gal(b1-4)GlcNAc(b1-",
      acceptor_alignment = "terminal",
      rejects = NULL,
      product = "GlcNAc(b1-"
    )
  ),
  type = "GH",
  species = "human"
)

# Create a custom enzyme with rejects
make_enzyme(
  name = "MySiaT",
  rules = list(
```

```
    list(
      acceptor = "Gal(b1-4)GlcNAc(b1-",
      acceptor_alignment = "terminal",
      rejects = c(
        # Reject a1-3 Fuc on GlcNAc
        "Fuc(a1-3)[Gal(b1-4)]GlcNAc(b1-",
        # Reject a1-2 Fuc on Gal
        "Fuc(a1-2)Gal(b1-4)GlcNAc(b1-"
      ),
      product = "Neu5Ac(a2-3)Gal(b1-4)GlcNAc(b1-"
    )
  ),
  type = "GT",
  species = "human"
)

# Create a custom enzyme with more than one rule
make_enzyme(
  name = "MySiaT",
  rules = list(
    list(
      acceptor = "Gal(b1-4)GlcNAc(b1-",
      acceptor_alignment = "terminal",
      rejects = NULL,
      product = "Neu5Ac(a2-3)Gal(b1-4)GlcNAc(b1-"
    ),
    # same acceptor, different product
    list(
      acceptor = "Gal(b1-4)GlcNAc(b1-",
      acceptor_alignment = "terminal",
      rejects = NULL,
      product = "Neu5Gc(a2-3)Gal(b1-4)GlcNAc(b1-"
    )
  ),
  type = "GT",
  species = "human"
)
```

`match_enzyme` *Match Residues Added by an Enzyme*

Description

This function finds residues in glycans that match the product motifs of a glycosyltransferase and returns their node indices.

Usage

```
match_enzyme(glycans, enzyme)
```

Arguments

glycans	A glyrepr::glycan_structure() vector.
enzyme	A glycosyltransferase enzyme() or a gene symbol for one. Glycoside hydrolases are not supported.

Value

A list of integer vectors with the same length as glycans. Each integer vector contains node indices for residues added by enzyme in the corresponding glycan.

Important notes

Here are some important notes for all functions in the glyenzy package.

Applicability:

All algorithms and enzyme information in glyenzy are applicable only to humans, and specifically to N-glycans and O-GalNAc glycans. Results may be inaccurate for other types of glycans (e.g., GAGs, glycolipids) or for glycans in other species (e.g., plants, insects).

Inclusiveness:

The algorithm takes an intentionally inclusive approach, assuming that all possible isoenzymes capable of catalyzing a given reaction may be involved. Therefore, results should be interpreted with caution.

For example, in humans, detection of the motif "Neu5Ac(a2-3)Gal(b1-" will return both "ST3GAL3" and "ST3GAL4". In reality, only one of them might be active, depending on factors such as tissue specificity.

Only "concrete" glycans:

The function only works for glycans containing **concrete** residues (e.g., "Glc", "GalNAc"), and not for glycans with **generic** residues (e.g., "Hex", "HexNAc").

Substituents:

Substituents (e.g. sulfation, phosphorylation) are not supported yet, and the algorithms might fail for glycans with substituents. If your glycans contain substituents, use glyrepr::remove_substituents() to get clean glycans.

Incomplete glycan structures:

If the glycan structure is incomplete or partially degraded, the result may be misleading.

Starting points:

- For N-glycans, the starting structure is assumed to be "Glc(3)Man(9)GlcNAc(2)", the N-glycan precursor transferred to Asn by OST.
- For O-GalNAc glycans, the starting structure is assumed to be "GalNAc(a1-".
- For O-GlcNAc glycans, the starting structure is assumed to be "GlcNAc(b1-".
- For O-Man glycans, the starting structure is assumed to be "Man(a1-".
- For O-Fuc glycans, the starting structure is assumed to be "Fuc(a1-".
- For O-Glc glycans, the starting structure is assumed to be "Glc(b1-".

Examples

```
glycan <- glyrepr::as_glycan_structure("Neu5Ac(a2-3)Gal(b1-3)GlcNAc(b1-")
match_enzyme(glycan, "ST3GAL3")
```

`path_biosynthesis` *Find a Biosynthesis Path Between Glycan Structures*

Description

Find a synthesis path from one glycan structure to another using enzymatic reactions. This function uses breadth-first search to find the shortest path or all possible paths within a given number of steps.

Usage

```
path_biosynthesis(from, to, enzymes = NULL, max_steps = 10, filter = NULL)
```

Arguments

`from`	A `glyrepr::glycan_structure()` scalar, or a character string supported by `glyparse::auto_parse()`. The starting glycan structure.
`to`	A `glyrepr::glycan_structure()` scalar, or a character string supported by `glyparse::auto_parse()`. The target glycan structure.
`enzymes`	A character vector of gene symbols, or a list of `enzyme()` objects. If `NULL` (default), all available enzymes will be used.
`max_steps`	Integer, maximum number of enzymatic steps to search. Default is 10.
`filter`	Optional function to filter generated glycans at each step. Should take a `glyrepr::glycan_structure()` vector as input and return a logical vector of the same length. It will be applied to all the generated glycans at each BFS step for pruning.

Value

An `igraph::igraph()` object representing the synthesis path(s). Vertices represent glycan structures with `name` attribute containing IUPAC-condensed strings. Edges represent enzymatic reactions with `enzyme` attribute containing gene symbols and `step` attribute indicating the step number.

Important notes

Here are some important notes for all functions in the `glyenzy` package.

Applicability:

All algorithms and enzyme information in glyenzy are applicable only to humans, and specifically to N-glycans and O-GalNAc glycans. Results may be inaccurate for other types of glycans (e.g., GAGs, glycolipids) or for glycans in other species (e.g., plants, insects).

Inclusiveness:

The algorithm takes an intentionally inclusive approach, assuming that all possible isoenzymes capable of catalyzing a given reaction may be involved. Therefore, results should be interpreted with caution.

For example, in humans, detection of the motif "Neu5Ac(a2-3)Gal(b1-" will return both "ST3GAL3" and "ST3GAL4". In reality, only one of them might be active, depending on factors such as tissue specificity.

Only "concrete" glycans:

The function only works for glycans containing **concrete** residues (e.g., `"Glc"`, `"GalNAc"`), and not for glycans with **generic** residues (e.g., `"Hex"`, `"HexNAc"`).

Substituents:

Substituents (e.g. sulfation, phosphorylation) are not supported yet, and the algorithms might fail for glycans with substituents. If your glycans contain substituents, use `glyrepr::remove_substituents()` to get clean glycans.

Incomplete glycan structures:

If the glycan structure is incomplete or partially degraded, the result may be misleading.

Starting points:

- For N-glycans, the starting structure is assumed to be "Glc(3)Man(9)GlcNAc(2)", the N-glycan precursor transferred to Asn by OST.
- For O-GalNAc glycans, the starting structure is assumed to be "GalNAc(a1-".
- For O-GlcNAc glycans, the starting structure is assumed to be "GlcNAc(b1-".
- For O-Man glycans, the starting structure is assumed to be "Man(a1-".
- For O-Fuc glycans, the starting structure is assumed to be "Fuc(a1-".
- For O-Glc glycans, the starting structure is assumed to be "Glc(b1-".

Examples

```
library(glyrepr)
library(glyparse)

# Find shortest path
from <- "Gal(b1-4)GlcNAc(b1-"
to <- "Neu5Ac(a2-6)Gal(b1-4)GlcNAc(b1-"
path <- path_biosynthesis(from, to, enzymes = "ST6GAL1", max_steps = 3)

# View the path
igraph::as_data_frame(path, what = "edges")
```

print.glyenzy_enzyme *Print method for glyenzy_enzyme objects*

Description

Print method for glyenzy_enzyme objects

Usage

```
## S3 method for class 'glyenzy_enzyme'
print(x, ...)
```

Arguments

x	A glyenzy_enzyme object.
...	Additional arguments passed to print methods.

trace_biosynthesis *Trace the Biosynthetic Path of Glycans*

Description

Reconstruct the biosynthetic pathway for one or more glycans using enzymatic reactions. This function uses a multi-target breadth-first search to find all feasible pathways that can synthesize all the target glycans.

Usage

```
trace_biosynthesis(glycans, enzymes = NULL, max_steps = 20, filter = NULL)
```

Arguments

glycans	A glyrepr::glycan_structure() vector, or a character vector of strings supported by glyparse::auto_parse(). Can also be a single glycan. If multiple glycans are provided, the starting structure will be decided by the first glycan. Therefore, please make sure glycans are not of mixed glycan types.
enzymes	A character vector of gene symbols, or a list of enzyme() objects. If NULL (default), all available enzymes will be used.
max_steps	Integer, maximum number of enzymatic steps to search. Default is 20.
filter	Optional function to filter generated glycans at each step. Should take a glyrepr::glycan_structure() vector as input and return a logical vector of the same length. It will be applied to all the generated glycans at each BFS step for pruning.

Value

An `igraph::igraph()` object representing the synthesis path(s). Vertices represent glycan structures with `name` attribute containing IUPAC-condensed strings. Edges represent enzymatic reactions with `enzyme` attribute containing gene symbols and `step` attribute indicating the step number. For multiple targets, the graph includes all synthesis paths needed to reach every target glycan.

Important notes

Here are some important notes for all functions in the `glyenzy` package.

Applicability:

All algorithms and enzyme information in glyenzy are applicable only to humans, and specifically to N-glycans and O-GalNAc glycans. Results may be inaccurate for other types of glycans (e.g., GAGs, glycolipids) or for glycans in other species (e.g., plants, insects).

Inclusiveness:

The algorithm takes an intentionally inclusive approach, assuming that all possible isoenzymes capable of catalyzing a given reaction may be involved. Therefore, results should be interpreted with caution.

For example, in humans, detection of the motif "Neu5Ac(a2-3)Gal(b1-" will return both "ST3GAL3" and "ST3GAL4". In reality, only one of them might be active, depending on factors such as tissue specificity.

Only "concrete" glycans:

The function only works for glycans containing **concrete** residues (e.g., `"Glc"`, `"GalNAc"`), and not for glycans with **generic** residues (e.g., `"Hex"`, `"HexNAc"`).

Substituents:

Substituents (e.g. sulfation, phosphorylation) are not supported yet, and the algorithms might fail for glycans with substituents. If your glycans contain substituents, use `glyrepr::remove_substituents()` to get clean glycans.

Incomplete glycan structures:

If the glycan structure is incomplete or partially degraded, the result may be misleading.

Starting points:

- For N-glycans, the starting structure is assumed to be "Glc(3)Man(9)GlcNAc(2)", the N-glycan precursor transferred to Asn by OST.
- For O-GalNAc glycans, the starting structure is assumed to be "GalNAc(a1-".
- For O-GlcNAc glycans, the starting structure is assumed to be "GlcNAc(b1-".
- For O-Man glycans, the starting structure is assumed to be "Man(a1-".
- For O-Fuc glycans, the starting structure is assumed to be "Fuc(a1-".
- For O-Glc glycans, the starting structure is assumed to be "Glc(b1-".

Examples

```
library(glyrepr)
library(glyparse)

# Rebuild the biosynthetic pathway of a single glycan
glycan <- "Neu5Ac(a2-3)Gal(b1-4)[Fuc(a1-3)]GlcNAc(b1-3)Gal(b1-3)GalNAc(a1-"
path <- trace_biosynthesis(glycan, max_steps = 20)

# Rebuild pathways for multiple glycans
glycans <- c(
  "Neu5Ac(a2-3)Gal(b1-4)[Fuc(a1-3)]GlcNAc(b1-3)Gal(b1-3)GalNAc(a1-",
  "Gal(b1-4)[Fuc(a1-3)]GlcNAc(b1-3)Gal(b1-3)GalNAc(a1-"
)
path <- trace_biosynthesis(glycans, max_steps = 20)

# View the path
igraph::as_data_frame(path, what = "edges")
```

---

view_enzyme *View Residues Added by an Enzyme*

Description

Visualize where an enzyme contributes residues to a glycan structure.

Usage

```
view_enzyme(glycan, enzyme)
```

Arguments

glycan	A `glyrepr::glycan_structure()`, or a glycan structure string supported by `glyparse::auto_parse()`.
enzyme	A glycosyltransferase `enzyme()` or a gene symbol for one. Glycoside hydrolases are not supported.

Details

`view_enzyme()` matches one `enzyme` against one `glycan` with the same matching rules used by `match_enzyme()`, then draws the glycan with the matched residues highlighted.

Value

A `ggplot` object returned by `glydraw::draw_cartoon()`. If no match is found, the glycan is drawn without highlighted residues and a cli alert is emitted.

Important notes

Here are some important notes for all functions in the `glyenzy` package.

Applicability:
All algorithms and enzyme information in glyenzy are applicable only to humans, and specifically to N-glycans and O-GalNAc glycans. Results may be inaccurate for other types of glycans (e.g., GAGs, glycolipids) or for glycans in other species (e.g., plants, insects).

Inclusiveness:
The algorithm takes an intentionally inclusive approach, assuming that all possible isoenzymes capable of catalyzing a given reaction may be involved. Therefore, results should be interpreted with caution.
For example, in humans, detection of the motif "Neu5Ac(a2-3)Gal(b1-" will return both "ST3GAL3" and "ST3GAL4". In reality, only one of them might be active, depending on factors such as tissue specificity.

Only "concrete" glycans:
The function only works for glycans containing **concrete** residues (e.g., `"Glc"`, `"GalNAc"`), and not for glycans with **generic** residues (e.g., `"Hex"`, `"HexNAc"`).

Substituents:
Substituents (e.g. sulfation, phosphorylation) are not supported yet, and the algorithms might fail for glycans with substituents. If your glycans contain substituents, use `glyrepr::remove_substituents()` to get clean glycans.

Incomplete glycan structures:
If the glycan structure is incomplete or partially degraded, the result may be misleading.

Starting points:

- For N-glycans, the starting structure is assumed to be "Glc(3)Man(9)GlcNAc(2)", the N-glycan precursor transferred to Asn by OST.
- For O-GalNAc glycans, the starting structure is assumed to be "GalNAc(a1-".
- For O-GlcNAc glycans, the starting structure is assumed to be "GlcNAc(b1-".
- For O-Man glycans, the starting structure is assumed to be "Man(a1-".
- For O-Fuc glycans, the starting structure is assumed to be "Fuc(a1-".
- For O-Glc glycans, the starting structure is assumed to be "Glc(b1-".

See Also

`match_enzyme()`, `glydraw::draw_cartoon()`

Examples

```
glycan <- glyrepr::as_glycan_structure("Neu5Ac(a2-3)Gal(b1-3)GlcNAc(b1-")

## Not run:
view_enzyme(glycan, "ST3GAL3")

## End(Not run)
```

Index