

Package: glyenzy (via r-universe)

October 22, 2026

Title Deconstruction Glycosylation Biosynthesis

Version 0.10.0

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, glycosidases, and sulfotransferases; 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, selected glycosphingolipid headgroups, and sulfate modifications in canonical N- and O-glycan contexts. Lipid aglycones are not represented, so glycolipid rules model their carbohydrate headgroups with ceramide omitted. 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 ggraph, knitr, rmarkdown, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

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

LinkingTo Rcpp, BH

SystemRequirements C++17

Imports Rcpp, checkmate, cli, glydraw (>= 0.8.0), glyexp (>= 0.15.0), glymotif (>= 0.19.0), glyrepr (>= 1.0.0), glyparse (>= 0.5.3), igraph, purrr, fastmap, dplyr, rlang, tibble, R6, ggplot2, S4Vectors, SummarizedExperiment

Depends R (>= 4.1)

LazyData true

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev zlib1g-dev

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

Date/Publication 2026-10-08 13:58:12 UTC

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

RemoteRef v0.10.0

RemoteSha 953ae7ac50792f7a4d8d63508531ca18049d5628

Contents

abstract_enzymes	3
apply_enzyme	5
autoplot.glyenzy_biosynthesis_network	7
count_enzyme	9
db_enzymes	11
enzyme	12
enzymes_from_rnaseq	14
find_enzyme	15
grow_glycans_step	17
have_enzyme	19
make_enzyme	21
match_enzyme	25
normalize_n_glycan	27
path_biosynthesis	30
path_biosynthesis_virtual	32
print.glyenzy_enzyme	34
product_substrate_ratio	35
trace_biosynthesis	36
trace_biosynthesis_virtual	39
view_enzyme	41

Index	44
--------------	-----------

Description

Return a separate collection of simplified N-glycan reaction rules. These enzymes represent reaction activities rather than individual genes. They are never included in `db_enzymes()`, `enzymes_from_rnaseq()`, or the default enzyme sets of other functions. Individual activities can also be retrieved by name with `enzyme()`, for example `enzyme("ManII")`.

Usage

```
abstract_enzymes(return_str = FALSE)
```

Arguments

<code>return_str</code>	If FALSE (default), return a named list of enzyme objects. If TRUE, return their activity names, which are accepted by <code>enzyme()</code> and by <code>enzyme</code> or <code>enzymes</code> arguments. An <code>enzymes</code> collection can use activity names or objects, but cannot mix abstract and concrete enzymes.
-------------------------	--

Details

The collection contains ManI, GnTI, ManII, GnTII, GnTIII, a6FucT, GnTIV, GnTV, b4GalT, iGnT, a3SiaT, and GlcH: 12 enzymes with 32 reaction rules. All have `glycan_type = "N"` and `species = "unspecified"`. The additional `localization` field records "cis", "medial", "trans", or "ER"; it is descriptive and does not impose compartment order during tracing.

The trimming activities follow the corresponding concrete enzymes in the ordinary database. ManI combines the 16 distinct whole-glycan rules of MAN1B1, MAN1A1, MAN1A2, and MAN1C1. ManII uses the shared MAN2A1/MAN2A2 rule, including its core alignment and rejection of bisecting GlcNAc or a galactosylated alpha1-3-arm GlcNAc. It removes either the alpha1-3 or alpha1-6 mannose first, preserving both Man4 intermediates. These rules give the same trimming reactions and network topology as their concrete counterparts; enzyme labels remain abstract.

Core transferase rules also follow the concrete database, including their acceptors, alignment, and rejects: GnTI uses MGAT1; GnTII uses MGAT2; GnTIII uses MGAT3; a6FucT uses FUT8; GnTIV uses MGAT4A/MGAT4B; and GnTV uses the N-glycan rule shared by MGAT5/MGAT5B. The O-mannose-specific MGAT5B rule is not included. GnTIII adds bisecting GlcNAc and has `localization = "medial"` in this abstract model.

FUT8, MGAT1, and MGAT2 have no explicit rejects in the concrete database; their acceptor and alignment restrictions are retained instead of adding new exclusions. GnTIII rejects galactosylated alpha1-3-arm antennae. GnTIV and GnTV retain their bisecting-GlcNAc and antenna-specific rejects. A reject excludes the matching reaction site, not every site on a glycan.

iGnT rejects the first LacNAc extension on both the beta1-2 and beta1-4 antennae of the alpha1-3 arm, without excluding sites on another arm. Because only iGnT initiates LacNAc extension in this collection, that arm remains unextended in networks generated from the default precursor.

The B3GNT4 reject for terminal Gal-GlcNAc-Gal-GlcNAc additionally prevents a second LacNAc extension. Thus the alpha1-6 arm can extend once, but not repeatedly through this activity.

a3SiaT retains the ST3GAL4/ST3GAL6 reject for alpha1-3 fucose on its Gal-GlcNAc acceptor. b4GalT has no explicit reject in the corresponding N-glycan B4GALT rules. Only restrictions for the represented N-glycan activity are imported; unrelated O-glycan and glycolipid reactions are excluded. These terminal activities retain the table acceptors, so this collection is not a replacement for the complete concrete database.

Ambiguous linkages use the existing motif-matching semantics, including potentially compatible reject matches. Product-motif statistics do not apply substrate rejects and do not establish that a product is reachable.

GlcH combines the three glucose-removal steps represented by MOGS and GANAB in the ordinary database. It removes one residue per step from the complete Glc3Man9GlcNAc2 precursor, retaining Glc2 and Glc1 intermediates. Thus the collection can trace from the usual `trace_biosynthesis()` starting structure without adding ordinary enzymes or changing the start.

Pass activities explicitly, for example `enzyme("ManI")` to `apply_enzyme()`, or activity names or the whole list from `abstract_enzymes()` to `trace_biosynthesis()`. Abstract reactions are explicit enzyme rules and have `is_virtual = FALSE` in biosynthesis networks. Existing restrictions on GH involvement statistics and residue matching still apply; this collection does not add GH support to those interfaces.

Value

A named list of `glyenzy_abstract_enzyme` objects, or a character vector when `return_str = TRUE`. Each object also inherits from `glyenzy_gt_enzyme` or `glyenzy_gh_enzyme`, and `glyenzy_enzyme`.

References

The simplified Golgi rules follow the user-supplied reaction table based on the N-glycosylation model of Spahn et al. (2016), with branch notation described by Liang et al. (2020). The supplied table takes precedence over differences in published versions, except for the concrete trimming and core-transferase rules, added GnTIII activity, and terminal-activity rejects described above. See the bundled rule-data attribution.

Spahn et al. (2016). A Markov chain model for N-linked protein glycosylation. [doi:10.1016/j.jymben.2015.10.007](https://doi.org/10.1016/j.jymben.2015.10.007).

Liang et al. (2020). A Markov model of glycosylation elucidates isozyme specificity and glycosyl-transferase interactions for glycoengineering. [doi:10.1016/j.crbiot.2020.01.001](https://doi.org/10.1016/j.crbiot.2020.01.001).

Examples

```
abstract_enzymes(return_str = TRUE)
enzyme("ManII")

man5 <- paste0(
  "Man(a1-3)[Man(a1-3)[Man(a1-6)]Man(a1-6)]",
  "Man(b1-4)GlcNAc(b1-4)GlcNAc(b1-4)"
)
apply_enzyme(man5, abstract_enzymes()[["GnTI"]])
trace_biosynthesis(man5, enzymes = abstract_enzymes(), max_virtual_steps = 0L)
```

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, structure_level = "intact")
```

Arguments

- | | |
|-----------------|---|
| glycans | A <code>glyrepr::glycan_structure()</code> , or a character vector of glycan structure strings supported by <code>glyparse::auto_parse()</code> . |
| enzyme | An enzyme object or a name accepted by <code>enzyme()</code> . |
| return_list | If NULL (default), return a list of <code>glyrepr::glycan_structure()</code> when glycans has length greater than 1, and a single <code>glyrepr::glycan_structure()</code> 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 <code>return_list = FALSE</code> and <code>length(glycans) > 1</code> , an error will be thrown. |
| structure_level | Output structure level. "intact" keeps the current behavior and returns products with exact linkages. "topological" removes linkages from products while preserving residue identities. "partial" classifies inputs with incomplete linkage or anomer information; it is not a generated-output option. The requested level cannot be lower-resolution than the input glycans. |

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:

Known-enzyme algorithms and enzyme information in glyenzy are applicable only to humans. Curated coverage is strongest for N-glycans and O-glycans and also includes selected glycosphingolipid headgroups and other glycan contexts. Lipid and protein aglycones are not represented, so glycolipid rules model the carbohydrate headgroup with ceramide omitted. Results may be inaccurate for unsupported glycan contexts or 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.

Concrete glycans by default:

Most functions only work for glycans containing **concrete** residues (e.g., "Glc", "GalNAc"), and not for glycans with **generic** residues (e.g., "Hex", "HexNAc"). Inputs with generic residues are supported where explicitly documented, such as `trace_biosynthesis()` and `path_biosynthesis()`.

Substituents:

Sulfate substituents are supported. Other substituents, such as phosphorylation and methylation, are not supported. Use `glyrepr::remove_substituents()` when unsupported substituents are present.

Incomplete or non-concrete glycan structures:

Some functions accept glycan structures that are incomplete, partially degraded, or contain generic residues. When accepted, glycans with a `glyrepr::get_structure_level()` other than "intact", or with a `glyrepr::get_mono_type()` other than "concrete", are matched with the lenient motif matching mode in `glymotif`. A warning is raised because enzyme predictions may be less reliable.

Starting points:

For known-enzyme path inference:

- 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-".
- For GlcCer glycans, the starting structure is assumed to be "Glc(b1-",
- For GalCer glycans, the starting structure is assumed to be "Gal(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")
```

```
autoplot.glyenzy_biosynthesis_network
```

Plot a glycan biosynthesis network

Description

autoplot.glyenzy_biosynthesis_network() draws the typed networks returned by [trace_biosynthesis\(\)](#), [trace_biosynthesis_virtual\(\)](#), [path_biosynthesis\(\)](#), and [path_biosynthesis_virtual\(\)](#). A layered DAG layout determines the node positions while accounting for converging biosynthesis routes. Candidate enzyme names stored on each reaction edge are combined into one label.

Usage

```
## S3 method for class 'glyenzy_biosynthesis_network'
autoplot(
  object,
  show_enzyme = TRUE,
  size = 0.4,
  node_gap = 0.25,
  level_gap = 0.6,
  width = 6,
  height = 6,
  units = c("in", "cm", "mm"),
  show_linkage = FALSE,
  orient = c("left", "right", "up", "down"),
  color_edge = FALSE,
  enzyme_label_style = c("condensed", "full"),
  highlight_target = NULL,
  ...
)
```

Arguments

object	A glyenzy_biosynthesis_network object returned by a biosynthesis function.
show_enzyme	Logical. Whether to label reaction edges with enzyme or virtual-enzyme names. Labels that would be too small to render after panel fitting are hidden with a warning.

size	Positive numeric whole-cartoon scale multiplier passed to <code>glydraw:geom_node_glycan()</code> . Defaults to 0.4.
node_gap	Non-negative physical clearance, in inches, between glycan nodes at the same rank.
level_gap	Non-negative physical clearance, in inches, between adjacent ranks. Increase this when enzyme labels are unusually long.
width, height	Positive, finite figure dimensions. Networks larger than the available figure are scaled proportionally, while smaller networks retain their natural size and are centered in the figure.
units	Units for width and height. One of "in" (the default), "cm", or "mm".
show_linkage	Logical. Whether glycan linkage annotations are shown. Defaults to FALSE for a compact network.
orient	Glycan drawing orientation passed to <code>glydraw:geom_node_glycan()</code> . One of "left", "right", "up", or "down"; the default is "left".
color_edge	Logical. Whether reaction edges and enzyme labels use the SNFG color of the residue added by each reaction. Defaults to FALSE, which draws them in dark grey. Concrete and virtual reactions use solid and dashed lines, respectively, in both modes. Reactions without one unambiguous added residue remain dark grey.
enzyme_label_style	How candidate enzymes on one reaction edge are labeled. "condensed" (the default) groups names with the same prefix and terminal number, for example, "B4GALT1/2/3, B3GALT3/4". "full" keeps complete names separated by " / ".
highlight_target	Logical or NULL. When TRUE, target glycans are drawn normally and all other glycans are semi-transparent. The default, NULL, highlights targets in multi-target networks but not in single-target networks. Reaction edges and enzyme labels are unchanged. This argument cannot be TRUE for networks returned by <code>path_biosynthesis()</code> or <code>path_biosynthesis_virtual()</code> .
...	Additional glycan appearance arguments accepted by <code>glydraw:glycanGrob()</code> , such as <code>node_size</code> , <code>colors</code> , or <code>style</code> .

Details

Glycan dimensions are measured before the network is laid out. Nodes at the same rank are separated by their rendered bounds, while rectangular edge caps stop arrows outside the source and target glycans. The plot uses a fixed-size panel inside a requested figure canvas so these clearances remain physical rather than changing with the coordinate range.

Concrete enzyme reactions use solid edge shafts, while virtual enzyme reactions use dashed shafts; arrowheads remain solid in both cases. This convention is documented rather than repeated in an in-plot legend.

Value

A `ggraph/ggplot` object with a fixed-size, collision-aware layered panel centered in the requested figure dimensions.

Figure size

width, height, and units describe the complete base figure returned by this method. Use the same dimensions with `ggplot2::ggsave()` or the corresponding knitr/Quarto figure options. Titles, legends, or margins added after this method returns may require additional output space.

Examples

```
## Not run:
network <- trace_biosynthesis_virtual(
  "GlcNAc(b1-4)Gal(b1-3)GalNAc(a1-"
)
network_plot <- ggplot2::autoplot(network, width = 8, height = 5)
ggplot2::ggsave(
  "biosynthesis-network.png",
  network_plot,
  width = 8,
  height = 5,
  units = "in"
)

## End(Not run)
```

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, method = c("motif", "path"))
```

Arguments

glycans	A <code>glyrepr::glycan_structure()</code> , or a character vector of glycan structure strings supported by <code>glyparse::auto_parse()</code> .
enzyme	An enzyme object or a name accepted by <code>enzyme()</code> .
method	Method used to count enzyme involvement. "motif" counts product motifs directly in each glycan. "path" counts enzyme-labeled edges in <code>trace_biosynthesis()</code> results, which is more accurate but slower.

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:

Known-enzyme algorithms and enzyme information in glyenzy are applicable only to humans. Curated coverage is strongest for N-glycans and O-glycans and also includes selected glycosphingolipid headgroups and other glycan contexts. Lipid and protein aglycones are not represented, so glycolipid rules model the carbohydrate headgroup with ceramide omitted. Results may be inaccurate for unsupported glycan contexts or 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.

Concrete glycans by default:

Most functions only work for glycans containing **concrete** residues (e.g., "Glc", "GalNAc"), and not for glycans with **generic** residues (e.g., "Hex", "HexNAc"). Inputs with generic residues are supported where explicitly documented, such as `trace_biosynthesis()` and `path_biosynthesis()`.

Substituents:

Sulfate substituents are supported. Other substituents, such as phosphorylation and methylation, are not supported. Use `glyrepr::remove_substituents()` when unsupported substituents are present.

Incomplete or non-concrete glycan structures:

Some functions accept glycan structures that are incomplete, partially degraded, or contain generic residues. When accepted, glycans with a `glyrepr::get_structure_level()` other than "intact", or with a `glyrepr::get_mono_type()` other than "concrete", are matched with the lenient motif matching mode in `glymotif`. A warning is raised because enzyme predictions may be less reliable.

Starting points:

For known-enzyme path inference:

- 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-".
- For GlcCer glycans, the starting structure is assumed to be "Glc(b1-",
- For GalCer glycans, the starting structure is assumed to be "Gal(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")

# Use reconstructed biosynthesis paths
count_enzyme(glycans, "ST6GAL1", method = "path")
```

db_enzymes

Get all enzymes

Description

Return a named list of ordinary built-in enzymes, or a character vector of gene symbols if `return_str` is TRUE. Abstract activities are available separately through [abstract_enzymes\(\)](#).

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	<i>Enzymes</i>
--------	----------------

Description

Use `enzyme()` with a gene symbol or an abstract name to load a predefined enzyme. For example, use `enzyme("ST3GAL3")` to load the enzyme ST3GAL3, and `enzyme("ManII")` for a general ManII (possibly MAN2A1 or MAN2A2). It returns an enzyme object that can be passed to any enzyme argument of all functions in this package. As a shortcut, passing a name is also valid, e.g. `have_enzyme(glycan, "ST3GAL3")`.

Usage

```
enzyme(symbol)
```

Arguments

`symbol` The gene symbol of an ordinary enzyme or the name of an abstract activity listed by `abstract_enzymes()`.

Value

A `glyenzy_enzyme` object. Abstract activities additionally inherit from `glyenzy_abstract_enzyme`.

Enzyme types (biological)

Enzymes can be classified by their biological rules. Three types of enzymes can be represented: glycosyltransferases (GTs), glycoside hydrolases (GHs), and sulfotransferases (STs).

- 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.
- STs add a sulfate substituent to an existing glycan residue.

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.

Enzyme types (computational)

There're also three types of enzymes you'll encounter in this package depending on how we define them, they are concrete enzymes, abstract enzymes, and virtual enzymes.

Concrete enzymes are the enzymes represented by gene symbols, e.g. B4GALT1. There can be many isoenzymes on this level. For example, B4GALT1/2/3/4/5 are all isoenzymes with similar substrate specificity.

Abstract enzymes are conventional enzymes without considering isoenzymes. For example, the above enzymes are described as b4GalT. Other examples include ManI, ManII, GnTI, etc. Currently only abstract enzymes for N-glycan biosynthesis are supported.

Both concrete and abstract enzymes can be used in enzyme or enzymes arguments throughout this package. A single enzymes collection must contain only concrete enzymes or only abstract enzymes; the two types cannot be mixed. Both types are also supported by the enzyme() function.

Another type of enzymes is called the virtual enzymes. You'll see them in the results of [path_biosynthesis_virtual\(\)](#) and [trace_biosynthesis_virtual\(\)](#). Find out more in the documentation of these two functions.

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
 - requires: optional OR-valued context motifs. When present, at least one requirement must match.
 - 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 STs, this is the product node corresponding to the modified acceptor residue. For GHs, this is NULL.
 - new_residue: the new residue added by a GT. For GHs and STs, this is NULL.
 - new_linkage: the linkage of the new residue added by a GT. For GHs and STs, this is NULL.
 - new_substituent: for STs, the sulfate token added by the rule, such as "6S". This field is absent for other enzyme types.
 - product_sub: for STs, the canonical complete substituent state of the modified product residue, such as "3S,6S". This field is absent for other enzyme types.
3. type: the broad type of the enzyme, "GT" for glycosyltransferase, "GH" for glycoside hydrolase, or "ST" for sulfotransferase. Starter GTs are encoded as type = "GT".
4. species: the species of the enzyme, e.g. "human" or "mouse".

5. glycan_type: NULL for enzymes that can act on all glycan classes, or one or more of "N", "O", "lipid", and "free" for class-specific enzymes. Structures ending in Glc or Gal without a represented carrier are compatible with both "lipid" and "free".

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

Examples

```
library(glyrepr)

enzyme("ST3GAL3")
enzyme("ManII")
```

enzymes_from_rnaseq *Filter enzymes using RNA-seq expression*

Description

enzymes_from_rnaseq() selects built-in enzymes whose genes are expressed at or above a TPM threshold. When mat contains multiple columns, the mean TPM across columns is used for each gene.

Usage

```
enzymes_from_rnaseq(mat, threshold = 1)
```

Arguments

mat	A numeric matrix of TPM values with gene symbols as unique row names and samples or replicates as columns.
threshold	A non-negative numeric scalar giving the minimum mean TPM for a gene to be considered expressed. The default follows Huang et al. (2021): genes with TPM below 1 are treated as not expressed or rarely expressed, whereas genes with TPM at least 1 are treated as expressed.

Value

A named list of `enzyme()` objects. Genes not represented in the built-in enzyme database are ignored.

References

Huang, Y.-F., Aoki, K., Akase, S., et al. (2021). Global mapping of glycosylation pathways in human-derived cells. *Developmental Cell*, 56, 1195-1209. doi:10.1016/j.devcel.2021.02.023.

Examples

```
tpm <- matrix(
  c(0.2, 1, 3),
  ncol = 1,
  dimnames = list(c("FUT8", "ST3GAL3", "MOGS"), "sample")
)
enzymes_from_rnaseq(tpm)
```

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, method = c("motif", "path"))
```

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.
method	Method used to infer enzyme involvement. "motif" checks product motifs directly in each glycan. "path" extracts enzymes from trace_biosynthesis() results, which is more accurate but slower.

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 glyzeny package.

Applicability:

Known-enzyme algorithms and enzyme information in glyzeny are applicable only to humans. Curated coverage is strongest for N-glycans and O-glycans and also includes selected glycosphingolipid headgroups and other glycan contexts. Lipid and protein aglycones are not represented, so glycolipid rules model the carbohydrate headgroup with ceramide omitted. Results may be inaccurate for unsupported glycan contexts or 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.

Concrete glycans by default:

Most functions only work for glycans containing **concrete** residues (e.g., "Glc", "GalNAc"), and not for glycans with **generic** residues (e.g., "Hex", "HexNAc"). Inputs with generic residues are supported where explicitly documented, such as `trace_biosynthesis()` and `path_biosynthesis()`.

Substituents:

Sulfate substituents are supported. Other substituents, such as phosphorylation and methylation, are not supported. Use `glyrepr::remove_substituents()` when unsupported substituents are present.

Incomplete or non-concrete glycan structures:

Some functions accept glycan structures that are incomplete, partially degraded, or contain generic residues. When accepted, glycans with a `glyrepr::get_structure_level()` other than "intact", or with a `glyrepr::get_mono_type()` other than "concrete", are matched with the lenient motif matching mode in `glymotif`. A warning is raised because enzyme predictions may be less reliable.

Starting points:

For known-enzyme path inference:

- 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-".
- For GlcCer glycans, the starting structure is assumed to be "Glc(b1-".
- For GalCer glycans, the starting structure is assumed to be "Gal(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-")

# Use reconstructed biosynthesis paths
find_enzyme(glycans, method = "path")
```

grow_glycans_step	<i>Grow Glycans with Enzymes</i>
-------------------	----------------------------------

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

<code>glycans</code>	A <code>glyrepr::glycan_structure()</code> , or a character vector of glycan structure strings supported by <code>glyparse::auto_parse()</code> .
<code>enzymes</code>	A character vector of concrete gene symbols or abstract activity names, or a list of <code>enzyme()</code> objects. Concrete and abstract enzymes cannot be mixed.
<code>n_steps</code>	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 glyzeny package.

Applicability:

Known-enzyme algorithms and enzyme information in glyzeny are applicable only to humans. Curated coverage is strongest for N-glycans and O-glycans and also includes selected glycosphingolipid headgroups and other glycan contexts. Lipid and protein aglycones are not represented, so glycolipid rules model the carbohydrate headgroup with ceramide omitted. Results may be inaccurate for unsupported glycan contexts or 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.

Concrete glycans by default:

Most functions only work for glycans containing **concrete** residues (e.g., "Glc", "GalNAc"), and not for glycans with **generic** residues (e.g., "Hex", "HexNAc"). Inputs with generic residues are supported where explicitly documented, such as `trace_biosynthesis()` and `path_biosynthesis()`.

Substituents:

Sulfate substituents are supported. Other substituents, such as phosphorylation and methylation, are not supported. Use `glyrepr::remove_substituents()` when unsupported substituents are present.

Incomplete or non-concrete glycan structures:

Some functions accept glycan structures that are incomplete, partially degraded, or contain generic residues. When accepted, glycans with a `glyrepr::get_structure_level()` other than "intact", or with a `glyrepr::get_mono_type()` other than "concrete", are matched with the lenient motif matching mode in `glymotif`. A warning is raised because enzyme predictions may be less reliable.

Starting points:

For known-enzyme path inference:

- 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-".
- For GlcCer glycans, the starting structure is assumed to be "Glc(b1-",
- For GalCer glycans, the starting structure is assumed to be "Gal(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(b1-4)"
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(b1-4)",
  "GlcNAc(b1-2)Man(a1-3)[GlcNAc(b1-2)Man(a1-6)]Man(b1-4)GlcNAc(b1-4)GlcNAc(b1-4)"
)
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, method = c("motif", "path"))
```

Arguments

glycans	A glyrepr::glycan_structure() , or a character vector of glycan structure strings supported by glyparse::auto_parse() .
enzyme	An enzyme object or a name accepted by enzyme() .
method	Method used to infer enzyme involvement. "motif" checks product motifs directly in each glycan. "path" extracts enzymes from trace_biosynthesis() results, which is more accurate but slower.

Value

A logical vector of the same length as glycans.

Important notes

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

Applicability:

Known-enzyme algorithms and enzyme information in glyzeny are applicable only to humans. Curated coverage is strongest for N-glycans and O-glycans and also includes selected glycosphingolipid headgroups and other glycan contexts. Lipid and protein aglycones are not represented, so glycolipid rules model the carbohydrate headgroup with ceramide omitted. Results may be inaccurate for unsupported glycan contexts or 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.

Concrete glycans by default:

Most functions only work for glycans containing **concrete** residues (e.g., "Glc", "GalNAc"), and not for glycans with **generic** residues (e.g., "Hex", "HexNAc"). Inputs with generic residues are supported where explicitly documented, such as `trace_biosynthesis()` and `path_biosynthesis()`.

Substituents:

Sulfate substituents are supported. Other substituents, such as phosphorylation and methylation, are not supported. Use `glyrepr::remove_substituents()` when unsupported substituents are present.

Incomplete or non-concrete glycan structures:

Some functions accept glycan structures that are incomplete, partially degraded, or contain generic residues. When accepted, glycans with a `glyrepr::get_structure_level()` other than "intact", or with a `glyrepr::get_mono_type()` other than "concrete", are matched with the lenient motif matching mode in glymotif. A warning is raised because enzyme predictions may be less reliable.

Starting points:

For known-enzyme path inference:

- 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-".
- For GlcCer glycans, the starting structure is assumed to be "Glc(b1-",
- For GalCer glycans, the starting structure is assumed to be "Gal(b1-"

Algorithm

The overall 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")

# Use reconstructed biosynthesis paths
have_enzyme(glycans, "ST6GAL1", method = "path")
```

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.

You can create regular glycosyltransferases (GTs), glycoside hydrolases (GHs), and sulfotransferases (STs) with this function. Special enzymes like starter GTs for glycosylation initiation are reserved for predefined enzymes.

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.

For sulfotransferases, the product must have the same residue and linkage topology as the acceptor and add exactly one sulfate substituent.

requires:

`requires` provides optional context motifs. Each entry contains `motif` and `alignment`; requirements are OR-valued, so at least one must match. An empty list means no additional context is required. Requirement motifs use sulfate-subset semantics, so an unsulfated requirement also matches the same topology with additional sulfates. Acceptor and reject motifs remain exact with respect to substituents during enzyme application.

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 or Modified by an Enzyme

Description

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

Usage

```
match_enzyme(glycans, enzyme, method = c("motif", "path"))
```

Arguments

glycans	A glyrepr::glycan_structure() vector.
enzyme	A glycosyltransferase or sulfotransferase enzyme() , or a name accepted by enzyme() . Glycoside hydrolases are not supported.
method	Method used to decide whether the enzyme is involved. "motif" matches product motifs directly in each glycan. "path" matches substrates and products from trace_biosynthesis() results back to each glycan, which is more accurate but slower.

Value

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

Important notes

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

Applicability:

Known-enzyme algorithms and enzyme information in glyenzy are applicable only to humans. Curated coverage is strongest for N-glycans and O-glycans and also includes selected glycosphingolipid headgroups and other glycan contexts. Lipid and protein aglycones are not represented, so glycolipid rules model the carbohydrate headgroup with ceramide omitted. Results may be inaccurate for unsupported glycan contexts or 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.

Concrete glycans by default:

Most functions only work for glycans containing **concrete** residues (e.g., "Glc", "GalNAc"), and not for glycans with **generic** residues (e.g., "Hex", "HexNAc"). Inputs with generic residues are supported where explicitly documented, such as `trace_biosynthesis()` and `path_biosynthesis()`.

Substituents:

Sulfate substituents are supported. Other substituents, such as phosphorylation and methylation, are not supported. Use `glyrepr::remove_substituents()` when unsupported substituents are present.

Incomplete or non-concrete glycan structures:

Some functions accept glycan structures that are incomplete, partially degraded, or contain generic residues. When accepted, glycans with a `glyrepr::get_structure_level()` other than "intact", or with a `glyrepr::get_mono_type()` other than "concrete", are matched with the lenient motif matching mode in `glymotif`. A warning is raised because enzyme predictions may be less reliable.

Starting points:

For known-enzyme path inference:

- 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-".
- For GlcCer glycans, the starting structure is assumed to be "Glc(b1-",
- For GalCer glycans, the starting structure is assumed to be "Gal(b1-"

Examples

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

normalize_n_glycan	<i>Normalize Topological N-Glycans</i>
--------------------	--

Description

Normalize selected Gal and Neu5Ac distributions in topological N-glycans. For the supported cases, glycans with the same numbers of antennae, Gal, and Neu5Ac are mapped to a common representative topology.

Usage

```
normalize_n_glycan(glycans)
```

Arguments

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

Value

A `glyrepr::glycan_structure()` vector of normalized glycans.

Why does this function exist?

`trace_biosynthesis()` and related functions support topological glycans. However, N-glycan annotations can assign residues to specific antennae even when the available data do not resolve those positions. These assignments can introduce unsupported distinctions into a biosynthesis network.

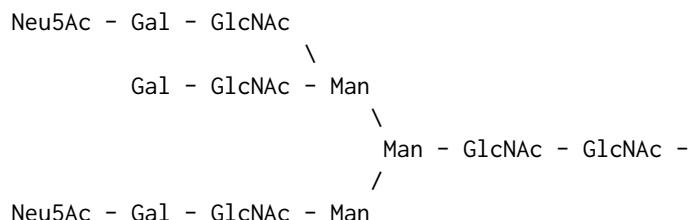
For example, consider the two glycans below.

Glycan 1:

```

Neu5Ac - Gal - GlcNAc
                        \
Neu5Ac - Gal - GlcNAc - Man
                        \
                        Man - GlcNAc - GlcNAc -
                        /
                        Gal - GlcNAc - Man
```

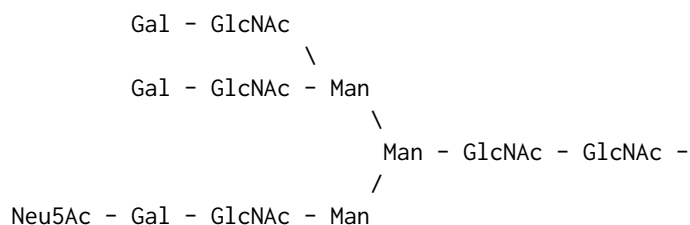
Glycan 2:



Both are tri-antennary glycans with two sialic acids. When mass spectrometry data do not resolve which antennae carry the sialic acids, either topology may be used to annotate the same unresolved structure. Nevertheless, tracing these annotations can produce different biosynthetic paths and introduce redundant branches when multiple glycans are traced together.

Glycan 3 below illustrates the problem. It has one sialic acid and could serve as a precursor to a disialylated glycan. Of the two topologies above, only Glycan 2 can be obtained from Glycan 3 by adding a single sialic acid. Tracing Glycan 1 and Glycan 3 together therefore requires separate paths rather than connecting them through a single sialylation step. When the sialic acid positions are unresolved, this distinction reflects the annotation rather than evidence for separate biosynthetic routes.

Glycan 3:

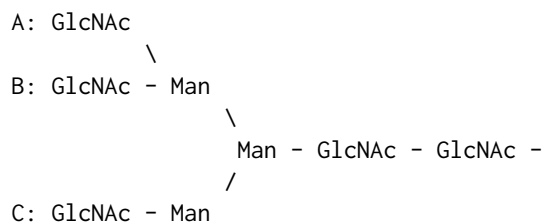


This function addresses the problem through a normalization convention. It applies rules to reposition Gal and Neu5Ac. In this example, mapping Glycan 1 to Glycan 2 allows the monosialylated Glycan 3 to connect to the disialylated representative through a single sialylation step. This convention does not imply that distinct isomers are biologically equivalent.

Algorithm

The algorithm repositions only Gal and Neu5Ac residues. To explain the algorithm, we name the branches for tri- and tetra-antennary glycans.

Tri-antennary:



Tetra-antennary:

```

A: GlcNAc
      \
B: GlcNAc - Man
          \
            Man - GlcNAc - GlcNAc -
          /
C: GlcNAc - Man
      /
D: GlcNAc

```

First, Gal residues are repositioned in the following cases:

- Three antennae, one Gal: positioned on C.
- Three antennae, two Gal: positioned on A and C.
- Four antennae, two Gal: positioned on A and C.

Next, Neu5Ac residues are repositioned in the following cases:

- Three antennae, two Gal, one Neu5Ac: positioned on the C Gal.
- Three antennae, three Gal, one Neu5Ac: positioned on the C Gal.
- Three antennae, three Gal, two Neu5Ac: positioned on the A and C Gal.
- Four antennae, three Gal, one Neu5Ac: with Gal on A, B, and C, Neu5Ac is positioned on the C Gal.
- Four antennae, three Gal, two Neu5Ac: with Gal on A, B, and C, Neu5Ac residues are positioned on the A and C Gal.
- Four antennae, four Gal, two Neu5Ac: positioned on the A and C Gal.

Other distributions are left unchanged. Branch labels describe topology, not linkage positions or drawing order. For three antennae, C is on the Man with one antenna; A and B are interchangeable. For four antennae, A/B and C/D are the two pairs. With three Gal, A/B is the pair with two Gal, and C is the occupied antenna of the other pair.

Only concrete, topological structures are accepted. Use `glyrepr::remove_linkages()` explicitly before calling this function on intact or partial structures. Missing values are preserved.

Only simple GlcNAc, Gal-GlcNAc, and Neu5Ac-Gal-GlcNAc antennae are normalized. Non-N-glycans, other antenna counts, and structures with extended or modified antennae, other sialic acids, or floating parts or substituents are returned unchanged. Core fucose on the reducing-end GlcNAc and a bisecting GlcNAc on the central Man are preserved.

Examples

```

x <- glyparse::auto_parse(paste0(
  "Gal(b1-4)GlcNAc(b1-2)[GlcNAc(b1-6)]Man(a1-6)",
  "[GlcNAc(b1-2)Man(a1-3)]Man(b1-4)GlcNAc(b1-4)GlcNAc(b1-6)"
))
normalize_n_glycan(glyrepr::remove_linkages(x))

```

path_biosynthesis	<i>Find a Biosynthesis Path Between Glycan Structures</i>
-------------------	---

Description

Find biosynthetic paths from one glycan structure to another. The default method uses known enzyme rules in a forward breadth-first search. To infer structure-driven paths without enzyme specificity, use `path_biosynthesis_virtual()`.

Usage

```
path_biosynthesis(
  from,
  to,
  enzymes = NULL,
  max_steps = 10,
  filter = NULL,
  max_virtual_steps = 0L
)
```

Arguments

from	A <code>glyrepr::glycan_structure()</code> scalar, or a character string supported by <code>glyparse::auto_parse()</code> . The starting glycan structure.
to	A <code>glyrepr::glycan_structure()</code> scalar, or a character string supported by <code>glyparse::auto_parse()</code> . The target glycan structure.
enzymes	A character vector of concrete gene symbols or abstract activity names, or a list of <code>enzyme()</code> objects. Concrete and abstract enzymes cannot be mixed. If NULL (default), all available concrete 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 <code>glyrepr::glycan_structure()</code> vector as input and return a logical vector of the same length. It filters generated products.
max_virtual_steps	Integer, maximum number of target-directed virtual enzyme steps allowed when no fully enzymatic path exists. Default is 0L, which disables virtual fallback. See the "Virtual fallback" section for more details.

Value

A `glyenzy_biosynthesis_network` object inheriting from `igraph::igraph()` and representing the synthesis path(s). Vertices represent glycan structures, with IUPAC-condensed strings in the name attribute and a logical target attribute indicating the target glycan. At most one directed edge connects each substrate and product. Every edge has an integer step, a logical `is_virtual`, a scalar display label in `enzyme`, and a list-valued `enzymes` attribute containing concrete enzyme candidates. For known reactions, `enzyme` combines the candidates with " / "; for virtual reactions,

it contains the structural virtual-enzyme name and enzymes is empty unless concrete candidates were annotated.

Important notes

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

Applicability:

Known-enzyme algorithms and enzyme information in glyenzy are applicable only to humans. Curated coverage is strongest for N-glycans and O-glycans and also includes selected glycosphingolipid headgroups and other glycan contexts. Lipid and protein aglycones are not represented, so glycolipid rules model the carbohydrate headgroup with ceramide omitted. Results may be inaccurate for unsupported glycan contexts or 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.

Concrete glycans by default:

Most functions only work for glycans containing **concrete** residues (e.g., "Glc", "GalNAc"), and not for glycans with **generic** residues (e.g., "Hex", "HexNAc"). Inputs with generic residues are supported where explicitly documented, such as `trace_biosynthesis()` and `path_biosynthesis()`.

Substituents:

Sulfate substituents are supported. Other substituents, such as phosphorylation and methylation, are not supported. Use `glyrepr::remove_substituents()` when unsupported substituents are present.

Incomplete or non-concrete glycan structures:

Some functions accept glycan structures that are incomplete, partially degraded, or contain generic residues. When accepted, glycans with a `glyrepr::get_structure_level()` other than "intact", or with a `glyrepr::get_mono_type()` other than "concrete", are matched with the lenient motif matching mode in glymotif. A warning is raised because enzyme predictions may be less reliable.

Starting points:

For known-enzyme path inference:

- 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-".
- For GlcCer glycans, the starting structure is assumed to be "Glc(b1-",
- For GalCer glycans, the starting structure is assumed to be "Gal(b1-"

Input compatibility

`trace_biosynthesis()`, `trace_biosynthesis_virtual()`, `path_biosynthesis()`, and `path_biosynthesis_virtual()` accept only glycans that share one monosaccharide type ("concrete" or "generic") and one structure level ("intact" or "topological"). Structures with mixed generic and concrete residues, partial structures, and missing structures are not supported. In the `path_*()` functions, `from` and `to` must have the same monosaccharide type and structure level.

Use `glyrepr::convert_to_generic()` to standardize monosaccharide types or `glyrepr::remove_linkages()` to standardize structures at the topological level.

Virtual fallback

Sometimes the biosynthesis network of a glycan cannot be fully resolved; i.e., some enzymatic steps are not inferred to be catalyzed by any known enzyme ("bad" steps). By default, an error is raised for these glycans.

`max_virtual_steps` provides a fallback for these glycans. For a "bad" step, a virtual enzyme is assigned to allow the algorithm to continue. For example, for the O-GalNAc core 5 "GalNAc(a1-3)GalNAc(a1-", an "a3GalNAcT" is assigned to the step that adds the a3 GalNAc. Unsupported sulfate additions similarly use "3SulfoT", "6SulfoT", or "?SulfoT".

Therefore, `max_virtual_steps` can also be interpreted as "the maximum number of glycosidic bonds or sulfate transfers that cannot be assigned by a known enzyme." Increasing this number loosens the criteria.

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")
```

path_biosynthesis_virtual

Find a Virtual Biosynthesis Path Between Glycan Structures

Description

Infer every structure-driven biosynthetic path from `from` to `to` by trimming terminal residues and sulfate groups from `to` backward to `from`. Unlike `path_biosynthesis()`, this does not require known enzyme rules.

Usage

```
path_biosynthesis_virtual(from, to, enzymes = NULL, annotate_enzymes = FALSE)
```

Arguments

from	A <code>glyrepr::glycan_structure()</code> scalar, or a character string supported by <code>glyparse::auto_parse()</code> . The starting glycan structure.
to	A <code>glyrepr::glycan_structure()</code> scalar, or a character string supported by <code>glyparse::auto_parse()</code> . The target glycan structure.
enzymes	A character vector of concrete gene symbols or abstract activity names, or a list of <code>enzyme()</code> objects. Concrete and abstract enzymes cannot be mixed. Used only when <code>annotate_enzymes</code> is TRUE; if NULL, all available concrete enzymes are considered.
annotate_enzymes	Whether to annotate each virtual transition with concrete enzymes whose rules can perform it. Defaults to FALSE.

Value

A `glyenzy_virtual_biosynthesis_network` object inheriting from `glyenzy_biosynthesis_network` and `igraph::igraph()`. Vertices contain IUPAC-condensed strings in `name` and a logical target attribute indicating the target glycan. At most one directed edge connects each substrate and product. Edges have a forward step, `is_virtual = TRUE`, a virtual-enzyme enzyme label, and a list-valued `enzymes` attribute. When `annotate_enzymes` is TRUE, `enzymes` contains every candidate concrete enzyme for each transition; otherwise each element is empty.

Virtual enzymes

Each edge is named for the residue added by that step. Intact glycans include the linkage anomer and acceptor position, so a beta-1,4-linked GlcNAc is labeled "b4GlcNAc". Topological glycans omit linkage information and use "GlcNAc". Generic topological glycans use their preserved generic residue names, such as "HexNAc".

Sulfation is represented as its own atomic transition. Sulfate additions at positions 3 and 6 use "3SulfoT" and "6SulfoT"; an unknown or other position uses "?SulfoT". A sulfated terminal residue is therefore desulfated before the residue itself can be trimmed.

Virtual tracing starts N-glycans at the N-glycan core and all other glycans at their reducing-end root residue. Sulfates are removed from these automatically selected starts. In `path_biosynthesis_virtual()`, the explicit from glycan is always the virtual starting structure, including any sulfate groups it contains; those sulfates must also occur in to. These networks represent structural possibilities rather than biological feasibility.

Generic structures do not retain glycan-class metadata. A generic structure matching the N-glycan-core topology is therefore assumed to be an N-glycan; use `path_biosynthesis_virtual()` with an explicit from when that topology belongs to another glycan class.

Input compatibility

trace_biosynthesis(), trace_biosynthesis_virtual(), path_biosynthesis(), and path_biosynthesis_virtual() accept only glycans that share one monosaccharide type ("concrete" or "generic") and one structure level ("intact" or "topological"). Structures with mixed generic and concrete residues, partial structures, and missing structures are not supported. In the path_*() functions, from and to must have the same monosaccharide type and structure level.

Use glyrepr::convert_to_generic() to standardize monosaccharide types or glyrepr::remove_linkages() to standardize structures at the topological level.

Examples

```
virtual_path <- path_biosynthesis_virtual(
  "Gal(b1-3)GalNAc(a1-",
  "GlcNAc(b1-4)Gal(b1-3)GalNAc(a1-"
)

annotated_path <- path_biosynthesis_virtual(
  "Gal(b1-3)GalNAc(a1-",
  "GlcNAc(b1-4)Gal(b1-3)GalNAc(a1-",
  annotate_enzymes = TRUE
)
```

```
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.

product_substrate_ratio

Calculate Product-Substrate Ratios

Description

Calculate the ratio between the motif quantification of enzyme products and substrates for glycosyltransferases and sulfotransferases. For glycoproteomics data, ratios are calculated independently for each glycosite.

Usage

```
product_substrate_ratio(exp, enzymes)
```

Arguments

exp	A <code>glyexp::GlycomicSE()</code> or <code>glyexp::GlycoproteomicSE()</code> object.
enzymes	A character vector of concrete gene symbols or abstract activity names for glycosyltransferases or sulfotransferases, or a list of their <code>enzyme()</code> objects. Concrete and abstract enzymes cannot be mixed. Starter glycosyltransferases are not supported because their substrates are not glycans.

Details

Each rule contributes its product and acceptor motif once. Quantifications are summed across all rules of an enzyme before division, including when multiple rules contain the same motif. Rule rejects and requires are ignored. Ratios with zero substrate quantification are returned as NA.

Motifs are matched strictly for intact glycan structures and leniently for partial or reduced structures, consistent with other glyenzy motif-based functions.

Value

A plain `SummarizedExperiment::SummarizedExperiment()` with a `product_substrate_ratio` assay. For glycomics data, `rowData()` contains enzyme. For glycoproteomics data, it contains enzyme, protein, and protein_site. `colData()` and metadata are preserved from exp.

References

Bao B, Kellman BP, Chiang AWT, et al. (2021). Correcting for sparsity and interdependence in glycomics by accounting for glycan biosynthesis. *Nature Communications*, 12, 4988. doi:[10.1038/s41467-021-25183-5](https://doi.org/10.1038/s41467-021-25183-5)

Examples

```
exp <- glyexp::real_experiment2[seq_len(10), seq_len(3)]
product_substrate_ratio(exp, "ST6GAL1")
```

trace_biosynthesis	<i>Trace the Biosynthetic Path of Glycans</i>
--------------------	---

Description

Reconstruct biosynthetic pathways for one or more glycans. The default method uses known enzyme rules in a forward, multi-target breadth-first search. To infer structure-driven paths without enzyme specificity, use `trace_biosynthesis_virtual()`.

Usage

```
trace_biosynthesis(
  glycans,
  enzymes = NULL,
  max_steps = NULL,
  filter = NULL,
  max_virtual_steps = 0L
)
```

Arguments

glycans	A <code>glyrepr::glycan_structure()</code> vector, or a character vector of strings supported by <code>glyparse::auto_parse()</code> . 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 concrete gene symbols or abstract activity names, or a list of <code>enzyme()</code> objects. Concrete and abstract enzymes cannot be mixed. If NULL (default), all available concrete enzymes will be used.
max_steps	Integer, maximum number of enzymatic steps to search, or NULL to infer it from the largest target glycan. The inferred value is the maximum monosaccharide and substituent count plus 4 for N-glycans, and the maximum count minus 1 for other glycans.
filter	Optional function to filter generated glycans at each step. Should take a <code>glyrepr::glycan_structure()</code> vector as input and return a logical vector of the same length. It filters generated products.
max_virtual_steps	Integer, maximum number of target-directed virtual enzyme steps allowed when no fully enzymatic path exists. Default is 0L, which disables virtual fallback. See the "Virtual fallback" section for more details.

Value

A `glyenzy_biosynthesis_network` object inheriting from `igraph::igraph()` and representing the synthesis path(s). Vertices represent glycan structures, with IUPAC-condensed strings in the name attribute and a logical target attribute indicating whether each vertex is a target glycan. At most one directed edge connects each substrate and product. Every edge has an integer step, a

logical `is_virtual`, a scalar display label in enzyme, and a list-valued enzymes attribute containing concrete enzyme candidates. For known reactions, enzyme combines the candidates with " / "; for virtual reactions, it contains the structural virtual-enzyme name and enzymes is empty unless concrete candidates were annotated.

For multiple targets, the graph includes all synthesis paths needed to reach every target glycan.

Input compatibility

`trace_biosynthesis()`, `trace_biosynthesis_virtual()`, `path_biosynthesis()`, and `path_biosynthesis_virtual()` accept only glycans that share one monosaccharide type ("concrete" or "generic") and one structure level ("intact" or "topological"). Structures with mixed generic and concrete residues, partial structures, and missing structures are not supported. In the `path_*()` functions, from and to must have the same monosaccharide type and structure level.

Use `glyrepr::convert_to_generic()` to standardize monosaccharide types or `glyrepr::remove_linkages()` to standardize structures at the topological level.

Virtual fallback

Sometimes the biosynthesis network of a glycan cannot be fully resolved; i.e., some enzymatic steps are not inferred to be catalyzed by any known enzyme ("bad" steps). By default, an error is raised for these glycans.

`max_virtual_steps` provides a fallback for these glycans. For a "bad" step, a virtual enzyme is assigned to allow the algorithm to continue. For example, for the O-GalNAc core 5 "GalNAc(a1-3)GalNAc(a1-", an "a3GalNAcT" is assigned to the step that adds the a3 GalNAc. Unsupported sulfate additions similarly use "3SulfoT", "6SulfoT", or "?SulfoT".

Therefore, `max_virtual_steps` can also be interpreted as "the maximum number of glycosidic bonds or sulfate transfers that cannot be assigned by a known enzyme." Increasing this number loosens the criteria.

Important notes

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

Applicability:

Known-enzyme algorithms and enzyme information in glyzeny are applicable only to humans. Curated coverage is strongest for N-glycans and O-glycans and also includes selected glycosphingolipid headgroups and other glycan contexts. Lipid and protein aglycones are not represented, so glycolipid rules model the carbohydrate headgroup with ceramide omitted. Results may be inaccurate for unsupported glycan contexts or 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.

Concrete glycans by default:

Most functions only work for glycans containing **concrete** residues (e.g., "Glc", "GalNAc"), and not for glycans with **generic** residues (e.g., "Hex", "HexNAc"). Inputs with generic residues are supported where explicitly documented, such as `trace_biosynthesis()` and `path_biosynthesis()`.

Substituents:

Sulfate substituents are supported. Other substituents, such as phosphorylation and methylation, are not supported. Use `glyrepr::remove_substituents()` when unsupported substituents are present.

Incomplete or non-concrete glycan structures:

Some functions accept glycan structures that are incomplete, partially degraded, or contain generic residues. When accepted, glycans with a `glyrepr::get_structure_level()` other than "intact", or with a `glyrepr::get_mono_type()` other than "concrete", are matched with the lenient motif matching mode in `glymotif`. A warning is raised because enzyme predictions may be less reliable.

Starting points:

For known-enzyme path inference:

- 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-".
- For GlcCer glycans, the starting structure is assumed to be "Glc(b1-",
- For GalCer glycans, the starting structure is assumed to be "Gal(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)

# 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)

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

`trace_biosynthesis_virtual`*Trace a Virtual Biosynthetic Path of Glycans*

Description

Reconstruct every structure-driven biosynthetic path for one or more glycans by trimming terminal residues and sulfate groups backward. Unlike `trace_biosynthesis()`, this does not require known enzyme rules.

Usage

```
trace_biosynthesis_virtual(glycans, enzymes = NULL, annotate_enzymes = FALSE)
```

Arguments

<code>glycans</code>	A <code>glyrepr::glycan_structure()</code> vector, or a character vector of strings supported by <code>glyparse::auto_parse()</code> . 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.
<code>enzymes</code>	A character vector of concrete gene symbols or abstract activity names, or a list of <code>enzyme()</code> objects. Concrete and abstract enzymes cannot be mixed. Used only when <code>annotate_enzymes</code> is TRUE; if NULL, all available concrete enzymes are considered.
<code>annotate_enzymes</code>	Whether to annotate each virtual transition with concrete enzymes whose rules can perform it. Defaults to FALSE.

Value

A `glyzeny_virtual_biosynthesis_network` object inheriting from `glyzeny_biosynthesis_network` and `igraph::igraph()`. Vertices contain IUPAC-condensed strings in `name` and a logical `target` attribute indicating whether each vertex is a target glycan. At most one directed edge connects each substrate and product. Edges have a `forward_step`, `is_virtual = TRUE`, a virtual-enzyme `enzyme_label`, and a list-valued `enzymes` attribute. When `annotate_enzymes` is TRUE, `enzymes` contains every candidate concrete enzyme for each transition; otherwise each element is empty.

Virtual enzymes

Each edge is named for the residue added by that step. Intact glycans include the linkage anomer and acceptor position, so a beta-1,4-linked GlcNAc is labeled "b4GlcNAcT". Topological glycans omit linkage information and use "GlcNAcT". Generic topological glycans use their preserved generic residue names, such as "HexNAcT".

Sulfation is represented as its own atomic transition. Sulfate additions at positions 3 and 6 use "3SulfoT" and "6SulfoT"; an unknown or other position uses "?SulfoT". A sulfated terminal residue is therefore desulfated before the residue itself can be trimmed.

Virtual tracing starts N-glycans at the N-glycan core and all other glycans at their reducing-end root residue. Sulfates are removed from these automatically selected starts. In `path_biosynthesis_virtual()`, the explicit from glycan is always the virtual starting structure, including any sulfate groups it contains; those sulfates must also occur in to. These networks represent structural possibilities rather than biological feasibility.

Generic structures do not retain glycan-class metadata. A generic structure matching the N-glycan-core topology is therefore assumed to be an N-glycan; use `path_biosynthesis_virtual()` with an explicit from when that topology belongs to another glycan class.

Important notes

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

Applicability:

Known-enzyme algorithms and enzyme information in glyzeny are applicable only to humans. Curated coverage is strongest for N-glycans and O-glycans and also includes selected glycosphingolipid headgroups and other glycan contexts. Lipid and protein aglycones are not represented, so glycolipid rules model the carbohydrate headgroup with ceramide omitted. Results may be inaccurate for unsupported glycan contexts or 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.

Concrete glycans by default:

Most functions only work for glycans containing **concrete** residues (e.g., "Glc", "GalNAc"), and not for glycans with **generic** residues (e.g., "Hex", "HexNAc"). Inputs with generic residues are supported where explicitly documented, such as `trace_biosynthesis()` and `path_biosynthesis()`.

Substituents:

Sulfate substituents are supported. Other substituents, such as phosphorylation and methylation, are not supported. Use `glyrepr::remove_substituents()` when unsupported substituents are present.

Incomplete or non-concrete glycan structures:

Some functions accept glycan structures that are incomplete, partially degraded, or contain generic residues. When accepted, glycans with a `glyrepr::get_structure_level()` other than "intact", or with a `glyrepr::get_mono_type()` other than "concrete", are matched with the lenient motif matching mode in `glymotif`. A warning is raised because enzyme predictions may be less reliable.

Starting points:

For known-enzyme path inference:

- 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-".
- For GlcCer glycans, the starting structure is assumed to be "Glc(b1-",
- For GalCer glycans, the starting structure is assumed to be "Gal(b1-"

Input compatibility

`trace_biosynthesis()`, `trace_biosynthesis_virtual()`, `path_biosynthesis()`, and `path_biosynthesis_virtual()` accept only glycans that share one monosaccharide type ("concrete" or "generic") and one structure level ("intact" or "topological"). Structures with mixed generic and concrete residues, partial structures, and missing structures are not supported. In the `path_*`() functions, from and to must have the same monosaccharide type and structure level.

Use `glyrepr::convert_to_generic()` to standardize monosaccharide types or `glyrepr::remove_linkages()` to standardize structures at the topological level.

Examples

```
library(glyrepr)
library(glyparse)

virtual_path <- trace_biosynthesis_virtual(
  "GlcNAc(b1-4)Gal(b1-3)GalNAc(a1-"
)

annotated_path <- trace_biosynthesis_virtual(
  "GlcNAc(b1-4)Gal(b1-3)GalNAc(a1-",
  annotate_enzymes = TRUE
)
```

view_enzyme

View Residues Added or Modified by an Enzyme

Description

Visualize where an enzyme adds or modifies residues in a glycan structure.

Usage

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

Arguments

glycan	A <code>glyrepr::glycan_structure()</code> , or a glycan structure string supported by <code>glyparse::auto_parse()</code> .
enzyme	A glycosyltransferase or sulfotransferase <code>enzyme()</code> , or a name accepted by <code>enzyme()</code> . 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:

Known-enzyme algorithms and enzyme information in `glyenzy` are applicable only to humans. Curated coverage is strongest for N-glycans and O-glycans and also includes selected glycosphingolipid headgroups and other glycan contexts. Lipid and protein aglycones are not represented, so glycolipid rules model the carbohydrate headgroup with ceramide omitted. Results may be inaccurate for unsupported glycan contexts or 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.

Concrete glycans by default:

Most functions only work for glycans containing **concrete** residues (e.g., "Glc", "GalNAc"), and not for glycans with **generic** residues (e.g., "Hex", "HexNAc"). Inputs with generic residues are supported where explicitly documented, such as `trace_biosynthesis()` and `path_biosynthesis()`.

Substituents:

Sulfate substituents are supported. Other substituents, such as phosphorylation and methylation, are not supported. Use `glyrepr::remove_substituents()` when unsupported substituents are present.

Incomplete or non-concrete glycan structures:

Some functions accept glycan structures that are incomplete, partially degraded, or contain generic residues. When accepted, glycans with a `glyrepr::get_structure_level()` other than "intact",

or with a `glyrepr::get_mono_type()` other than "concrete", are matched with the lenient motif matching mode in `glymotif`. A warning is raised because enzyme predictions may be less reliable.

Starting points:

For known-enzyme path inference:

- 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-".
- For GlcCer glycans, the starting structure is assumed to be "Glc(b1-",
- For GalCer glycans, the starting structure is assumed to be "Gal(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

abstract_enzymes, 3
abstract_enzymes(), 11, 12
apply_enzyme, 5
apply_enzyme(), 4
autoplot.glyenzy_biosynthesis_network,
7

count_enzyme, 9

db_enzymes, 11
db_enzymes(), 3

enzyme, 12
enzyme(), 3, 5, 9, 12, 14, 17, 19, 21, 23, 25,
30, 33, 35, 36, 39, 42
enzymes_from_rnaseq, 14
enzymes_from_rnaseq(), 3

find_enzyme, 15

ggplot2::ggsave(), 9
glydraw::draw_cartoon(), 42, 43
glydraw::geom_node_glycan(), 8
glydraw::glycanGrob(), 8
glyexp::GlycomicSE(), 35
glyexp::GlycoproteomicSE(), 35
glymotif::have_motif(), 22, 23
glyparse::auto_parse(), 5, 9, 15, 17, 19,
22, 27, 30, 33, 36, 39, 42
glyrepr::convert_to_generic(), 32, 34,
37, 41
glyrepr::get_mono_type(), 6, 10, 16, 18,
20, 26, 31, 38, 40, 43
glyrepr::get_structure_level(), 6, 10,
16, 18, 20, 26, 31, 38, 40, 42
glyrepr::glycan_structure(), 5, 9, 15,
17–19, 22, 25, 27, 30, 33, 36, 39, 42
glyrepr::remove_linkages(), 29, 32, 34,
37, 41
glyrepr::remove_substituents(), 6, 10,
16, 18, 20, 26, 31, 38, 40, 42

grow_glycans (grow_glycans_step), 17
grow_glycans_step, 17

have_enzyme, 19

igraph::igraph(), 30, 33, 36, 39

make_enzyme, 21
match_enzyme, 25
match_enzyme(), 42, 43

normalize_n_glycan, 27

path_biosynthesis, 30
path_biosynthesis(), 7, 8, 32, 34, 37, 41
path_biosynthesis_virtual, 32
path_biosynthesis_virtual(), 7, 8, 13, 30,
32–34, 37, 40, 41
print.glyenzy_enzyme, 34
product_substrate_ratio, 35

SummarizedExperiment::SummarizedExperiment(),
35

trace_biosynthesis, 36
trace_biosynthesis(), 4, 7, 9, 15, 19, 25,
27, 39
trace_biosynthesis_virtual, 39
trace_biosynthesis_virtual(), 7, 13, 32,
34, 36, 37, 41

view_enzyme, 41