Package: glyanno (via r-universe)

June 3, 2026

Title Hierarchical Glycan Annotation

Version 0.5.0

Description Hierarchical annotation of glycans based on molecule mass, composition, and structure. Starting from a molecule mass, glyanno calculates possible glycan compositions. Given a glycan composition, glyanno further deduces possible glycan structures. For glycan structures with generic monosaccharides (e.g., ``Hex", ``HexNAc"), glyanno refines them into specific types (e.g., ``Glc", ``Gal"). For structures lacking linkage information (e.g., ``Gal(??-?)GalNAc(??-"), glyanno infers the most likely linkages (e.g., ``Gal(b1-3)GalNAc(a1-").

License MIT + file LICENSE

Suggests 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/glyanno/

Imports checkmate, cli, dplyr, glue, glyparse (>= 0.5.3), glyrepr (>= 0.11.0), glymotif, glydb (>= 0.4.0), igraph, purrr, rlang, stringr, tidyselect, tidyr, tibble, httr2

VignetteBuilder knitr

Depends R (>= 4.1)

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

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

Date/Publication 2026-04-30 10:20:45 UTC

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

RemoteRef v0.5.0

RemoteSha b5308bd8b63947953001c131188af172103b4fcc

Contents

`calculate_mz` *Calculate m/z values of glycans*

Description

This function calculates m/z values from glycans. Different adducts, mass types, and derivatives are supported. Custom mass dictionaries are also supported.

Usage

```
calculate_mz(glycans, charge = 1, adduct = "H+", mass_dict = NULL, safe = TRUE)
```

Arguments

`glycans`	Glycans to calculate m/z values from. Valid inputs include: • A `glyrepr::glycan_structure()` vector. • A `glyrepr::glycan_composition()` vector. • Byonic style composition strings (e.g. Hex(5)HexNAc(2)). • Simple style composition strings (e.g. H5N4F1S1). • Any structure strings supported by `glyparse::auto_parse()`.
`charge`	Charge to use. Can be 0, 1, 2, 3, -1, -2, -3, etc. 0 means neutral. Default is 1.
`adduct`	Adduct to use. Can be "H+", "K+", "Na+", "NH4+", "H-", "Cl-", "HCO3-". Default is "H+". • When `charge` is 0, `adduct` is ignored. • When `charge` is positive, `adduct` can only be "H+", "K+", "Na+", "NH4+". • When `charge` is negative, `adduct` can only be "H-", "Cl-", "HCO3-".
`mass_dict`	A named numeric vector of the mass of each monosaccharide residue. Default is `glyanno_mass_dict(deriv = "none", mass_type = "mono")`. If a custom mass dictionary is provided, please make sure the names of the vector are the same as the names in `glyanno_mass_dict()`.
`safe`	Whether to raise an error when unsupported monosaccharides or substituents are found.

- If TRUE (default), an error will be raised.
- If FALSE, a warning will be raised and m/z values for invalid glycans will be set to NA.

Value

A numeric vector of m/z values.

See Also

glyanno_mass_dict()

Examples

```
library(glyrepr)

# Different input types
calculate_mz(glycan_composition(c(Gal = 1, GalNAc = 1)), charge = 0)
calculate_mz("Gal(1)GalNAc(1)", charge = 0)
calculate_mz(as_glycan_structure("Gal(b1-3)GalNAc(a1-"), charge = 0)
calculate_mz("Gal(b1-3)GalNAc(a1-", charge = 0)

# For common situation in MALDI-TOF MS
calculate_mz("Man(5)GlcNAc(2)", charge = 1,adduct = "Na+")

# For common situation in ESI MS
calculate_mz("Man(5)GlcNAc(2)", charge = 1, adduct = "H+")

# Calculate permethylated m/z values
calculate_mz("Man(5)GlcNAc(2)", charge = 0, mass_dict = glyanno_mass_dict(deriv = "permethyl"))

# Calculate average mass
calculate_mz("Man(5)GlcNAc(2)", charge = 0, mass_dict = glyanno_mass_dict(mass_type = "average"))

# Vectorization
calculate_mz(c("Man(5)GlcNAc(2)", "Gal(1)GalNAc(1)"), charge = 1, adduct = "Na+")
```

---

comp_to_struc *Convert glycan composition to glycan structure*

Description

Given glycan compositions, this function matches them to all possible glycan structures in the glydb database.

Usage

```
comp_to_struc(comps, db = NULL, return_best = FALSE)
```

Arguments

comps	Glycan compositions to match against. Can be either: • A glyrepr::glycan_composition() vector. • Byonic style composition strings (e.g. Hex(5)HexNAc(2)). • Simple style composition strings (e.g. H5N4F1S1).
db	Glycan structures to match against. Can be a glyrepr::glycan_structure() vector or any structure strings supported by glyparse::auto_parse(). If not provided, glydb::glydb_structures(structure_level = "intact") will be used.
return_best	If TRUE, only return the highest confidence match for each composition. Requires db to have a confidence attribute. Use glydb::glydb_structures() for db to enable this feature. Default is FALSE.

Value

If return_best=TRUE: A glyrepr::glycan_structure() vector with the same length as comps. Unmatched compositions are returned as NA. If return_best=FALSE: A tibble with the following columns:

- composition: The glycan compositions, as glyrepr::glycan_composition() vector.
- structure: The possible glycan structures, as glyrepr::glycan_structure() vector. Note that one glycan composition can have multiple rows in the result, corresponding to different possible glycan structures.

How to set db

The db parameter is very important for all functions in this package. By default, it uses all available glycans in the glydb package, which is usually larger than what you need. You can use helper functions in glydb to narrow down the database, e.g. glydb::glydb_compositions() or glydb::glydb_structures().

You can use the species and glycan_type parameters to focus on specific species and glycan type. For example, if you are only interested in N-glycan compositions in human, you can use glydb::glydb_compositions(species = "Homo sapiens", glycan_type = "N"). Also, you can decide the level of information in the database by setting mono_type of glydb::glydb_compositions() and structure_level of glydb::glydb_structures().

You can then pass the result to the db parameter of this function. For example,

```
my_db <- glydb::glydb_compositions(species = "Homo sapiens", glycan_type = "N")
mz_to_comp(mz, db = my_db)
```

See Also

glyparse::auto_parse()

Examples

```
comp_to_struc("H5N2")
```

enhance_comp *Enhance glycan composition*

Description

Given a generic glycan composition (e.g. Hex(5)HexNAc(2)), this function gives all possible concrete glycan compositions (e.g. Man(5)GlcNAc(2)).

Usage

```
enhance_comp(comps, db = NULL, return_best = FALSE)
```

Arguments

comps	A `glyrepr::glycan_composition()` vector, or a character vector of glycan composition strings of Byonic or simple style (e.g. "Hex(5)HexNAc(2)", "H5N4F1S1"). Generic compositions (e.g. Hex(5)HexNAc(2)) will be matched to all possible concrete compositions in db. Concrete compositions (e.g. Man(5)GlcNAc(2)) will be returned as is.
db	A `glydb::glydb_compositions()` vector, or a character vector of glycan composition strings of Byonic or simple style (e.g. "Man(5)GlcNAc(2)", "H5N4F1S1"). All compositions in db must be concrete (e.g. Man(5)GlcNAc(2)). If not provided, `glydb::glydb_compositions(mono_type = "concrete")` will be used.
return_best	Logical. If TRUE, only return the highest confidence match for each input composition. Requires db to have a confidence attribute. Use `glydb::glydb_compositions()` for db to enable this feature. Defaults to FALSE.

Value

If `return_best=TRUE`: A `glyrepr::glycan_composition()` vector with the same length as comps. Unmatched compositions are returned as NA. If `return_best=FALSE`: A tibble with the following columns:

- raw: The original compositions.
- enhanced: The enhanced compositions. Note that one raw composition can have different enhanced compositions as multiple rows in the result.

How to set db

The db parameter is very important for all functions in this package. By default, it uses all available glycans in the glydb package, which is usually larger than what you need. You can use helper functions in glydb to narrow down the database, e.g. `glydb::glydb_compositions()` or `glydb::glydb_structures()`.

You can use the species and glycan_type parameters to focus on specific species and glycan type. For example, if you are only interested in N-glycan compositions in human, you can use `glydb::glydb_compositions(species = "Homo sapiens", glycan_type = "N")`. Also, you can

decide the level of information in the database by setting `mono_type` of `glydb::glydb_compositions()` and `structure_level` of `glydb::glydb_structures()`.

You can then pass the result to the `db` parameter of this function. For example,

```
my_db <- glydb::glydb_compositions(species = "Homo sapiens", glycan_type = "N")
mz_to_comp(mz, db = my_db)
```

Examples

```
enhance_comp("Hex(5)HexNAc(2)")
```

`enhance_struc` *Enhance glycan structure*

Description

Given a glycan structure vector of any resolution level (see `glyrepr::get_structure_level()` for details), this function gives all possible glycan structures of higher resolution level.

Usage

```
enhance_struc(strucs, db = NULL, return_best = FALSE)
```

Arguments

`strucs`	A `glyrepr::glycan_structure()` vector, or a character vector of glycan structure strings supported by `glyparse::auto_parse()`.
`db`	A `glydb::glydb_structures()` vector, or a character vector of glycan structure strings supported by `glyparse::auto_parse()`. If not provided, a default structure vector is loaded from `glydb::glydb_structures()` at "intact" level. If `db` has a lower or equal resolution level than `strucs`, the result will be the same as `strucs` (no enhancement).
`return_best`	Logical. If `TRUE`, only return the best matching structure (highest confidence) for each input structure. Requires `db` to have a `confidence` attribute. Use `glydb::glydb_structures()` for `db` to enable this feature. Default is `FALSE`.

Details

The target resolution level is determined from `db`. When `db` is NULL, the default `glydb::glydb_structures()` at "intact" level is used.

Value

If `return_best=TRUE`: An unnamed `glyrepr::glycan_structure()` vector with the same length as `strucs`. Unmatched structures are returned as `NA`. If `return_best=FALSE`: A tibble with the following columns:

- `raw`: The original glycan structures.
- `enhanced`: The enhanced glycan structures. Note that one `raw` glycan structure can have different `enhanced` glycan structures as multiple rows in the result.

Examples

```
# From topological level to intact level
db_intact <- c("Gal(b1-3)GalNAc(a1-", "Gal(b1-4)GalNAc(a1-")
enhance_struc("Gal(??-?)GalNAc(??-", db = db_intact)

# From basic level to topological level
db_topo <- "Gal(??-?)GalNAc(??-"
enhance_struc("Hex(??-?)HexNAc(??-", db = db_topo)

# From partial level to intact level
enhance_struc("Gal(b1-?)GalNAc(a1-", db = db_intact)
```

`fill_anomer_pos` *Fill anomer positions*

Description

Add anomer positions to glycan structures with missing anomer information based on biological knowledge. For example, "Gal(??-?)GalNAc(??-" will be converted to "Gal(?1-?)GalNAc(?1-". For anomer positions that are already specified in the input structures, this function will not modify them.

Usage

```
fill_anomer_pos(strucs)
```

Arguments

`strucs`	A `glyrepr::glycan_structure()` vector of "concrete" monosaccharides.

Details

This function is intended to be used by `struc_to_glytoucan()` to ensure that glycan structures have complete anomer information before attempting to inquire the GlyTouCan database.

For these monosaccharides, the anomer position is "2": "Neu5Ac", "Neu5Gc", "Neu", "Kdn", "Pse", "Leg", "Aci", "4eLeg", "Kdo", "Dha", "Fru", "Tag", "Sor", and "Psi". All other monosaccharides are assumed to have anomer positions on "1".

Value

A `glyrepr::glycan_structure()` vector with anomer positions added where missing.

Examples

```
library(glyrepr)
glycans <- as_glycan_structure(c(
  "Gal(??-?)GalNAc(??-",
  "Neu5Ac(??-?)Gal(??-?)GalNAc(??-"
))
fill_anomer_pos(glycans)
```

---

glyanno_mass_dict *Default Mass Dictionary for Glycan Residues*

Description

A named numeric vector of the mass of each monosaccharide residue. Used by default as the `mass_dict` argument in `calculate_mz()`.

The names includes:

- Monosaccharides: Hex, HexNAc, dHex, dHexNAc, ddHex, Pen, HexA, HexN, NeuAc, NeuGc, Kdn, Neu
- Ions: H+, H, H2O, K+, Na+, NH4+, H-, Cl-, HCO3-
- Substituents: S, P
- Reducing end: red_end, which is the additional mass of the reducing end of the glycan.

Usage

```
glyanno_mass_dict(deriv = "none", mass_type = "mono")
```

Arguments

`deriv`	A character scalar of the derivatization method to use. Can be "none", "permethyl", or "peracetyl". Default is "none".
`mass_type`	"mono" for monoisotopic mass, "average" for average mass. Default is "mono".

Value

A named numeric vector.

Examples

```
glyanno_mass_dict(deriv = "none", mass_type = "mono")
glyanno_mass_dict(deriv = "permethyl", mass_type = "mono")
glyanno_mass_dict(deriv = "none", mass_type = "average")
```

mz_to_comp *Convert m/z values to glycan composition*

Description

Given m/z values, this function matches them to all possible glycan compositions in the glydb database.

Usage

```
mz_to_comp(
  mz,
  tol = ppm(10),
  db = NULL,
  return_best = FALSE,
  charge = 1,
  adduct = "H+",
  mass_dict = NULL
)
```

Arguments

mz	A numeric vector of m/z values.
tol	A numeric scalar of the tolerance for the m/z value in Da or a ppm() object for dynamic tolerance. Default is ppm(10).
db	Glycan compositions to match against. Can be a glyrepr::glycan_composition() vector or glycan composition strings in Byonic style (e.g. Hex(5)HexNAc(2)) or simple style (e.g. H5N4F1S1). If not provided, glydb::glydb_compositions(mono_type = "concrete") will be used.
return_best	A logical scalar. If TRUE, only the match with the highest confidence score is returned for each m/z value. The db must have a confidence attribute. Use glydb::glydb_compositions() for db to enable this feature. Default is FALSE.
charge	Charge to use. Can be 0, 1, 2, 3, -1, -2, -3, etc. 0 means neutral. Default is 1.
adduct	Adduct to use. Can be "H+", "K+", "Na+", "NH4+", "H-", "Cl-", "HCO3-". Default is "H+". • When charge is 0, adduct is ignored. • When charge is positive, adduct can only be "H+", "K+", "Na+", "NH4+". • When charge is negative, adduct can only be "H-", "Cl-", "HCO3-".
mass_dict	A named numeric vector of the mass of each monosaccharide residue. Default is glyanno_mass_dict(deriv = "none", mass_type = "mono"). If a custom mass dictionary is provided, please make sure the names of the vector are the same as the names in glyanno_mass_dict().

Value

If `return_best=TRUE`: An unnamed `glyrepr::glycan_composition()` vector with the same length as `mz`. Unmatched m/z values are returned as `NA`. If `return_best=FALSE`: A tibble with the following columns:

- `mz`: The molecule m/z values, same as the input `mz`.
- `composition`: The possible glycan compositions, as `glyrepr::glycan_composition()` vector. Note that one m/z value can have multiple rows in the result, corresponding to different possible glycan compositions.

How to set `db`

The `db` parameter is very important for all functions in this package. By default, it uses all available glycans in the `glydb` package, which is usually larger than what you need. You can use helper functions in `glydb` to narrow down the database, e.g. `glydb::glydb_compositions()` or `glydb::glydb_structures()`.

You can use the `species` and `glycan_type` parameters to focus on specific species and glycan type. For example, if you are only interested in N-glycan compositions in human, you can use `glydb::glydb_compositions(species = "Homo sapiens", glycan_type = "N")`. Also, you can decide the level of information in the database by setting `mono_type` of `glydb::glydb_compositions()` and `structure_level` of `glydb::glydb_structures()`.

You can then pass the result to the `db` parameter of this function. For example,

```
my_db <- glydb::glydb_compositions(species = "Homo sapiens", glycan_type = "N")
mz_to_comp(mz, db = my_db)
```

See Also

`ppm()`, `glyanno_mass_dict()`

Examples

```
mz_to_comp(933.3175, charge = 1, adduct = "Na+")
```

`ppm` *Calculate PPM*

Description

This helper function should be used in `mz_to_comp()` as the `tol` argument. It makes the function using PPM as dynamic tolerance.

Usage

```
ppm(x)
```

Arguments

x	A numeric scalar of the PPM value.

Value

A function that takes a numeric vector of m/z values and returns a numeric vector of tolerances.

Examples

```
ppm(10)(2368.84)
ppm(10)(c(2368.84, 2368.85))
```

---

struc_to_glytoucan *Assign GlyTouCan accessions to glycan structures*

Description

This function takes a vector of glycan structures and returns the corresponding GlyTouCan accessions. Under the hood, it uses the GlycanFormatConverter API maintained by the Glycosmos project.

Usage

```
struc_to_glytoucan(strucs)
```

Arguments

strucs	A `glyrepr::glycan_structure()` vector, or a character vector of glycan text representations supported by `glyparse::auto_parse()`. The glycan structure must have "concrete" monosaccharides (e.g., Gal, GalNAc).

Details

For "topological" structures (e.g., "Gal(??-?)GalNAc(??-"), this function will first call `fill_anomer_pos()` to fill in the missing anomeric positions before querying the API. This is necessary because all glycan structures in GlyTouCan must have defined anomeric positions.

Value

A character vector of GlyTouCan accessions corresponding to the input glycan structures. If a structure cannot be converted to a GlyTouCan accession, the corresponding entry will be `NA`.

Index