

Package ‘glyparse’

November 4, 2025

Title Parsing Glycan Structure Text Representations

Version 0.5.3

Description Provides functions to parse glycan structure text representations into 'glyrepr' glycan structures. Currently, it supports StrucGP-style, pGlyco-style, IUPAC-condensed, IUPAC-extended, IUPAC-short, WURCS, Linear Code, and GlycoCT format. It also provides an automatic parser to detect the format and parse the structure string.

License MIT + file LICENSE

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

URL <https://glycoverse.github.io/glyparse/>,
<https://github.com/glycoverse/glyparse>

Imports checkmate, cli, dplyr, glyrepr (>= 0.7.0), igraph, purrr,
rlang, rstackdeque, stringr

Depends R (>= 4.1)

VignetteBuilder knitr

BugReports <https://github.com/glycoverse/glyparse/issues>

NeedsCompilation no

Author Bin Fu [aut, cre, cph] (ORCID: <<https://orcid.org/0000-0001-8567-2997>>)

Maintainer Bin Fu <23110220018@m.fudan.edu.cn>

Repository CRAN

Date/Publication 2025-11-04 19:30:02 UTC

Contents

auto_parse	2
parse_glycoct	3

parse_iupac_condensed	4
parse_iupac_extended	5
parse_iupac_short	6
parse_linear_code	7
parse_pglyco_struc	7
parse_strucgp_struc	8
parse_wurcs	8
Index	10

auto_parse	<i>Automatic Structure Parsing</i>
------------	------------------------------------

Description

Detect the structure string type and use the appropriate parser to parse automatically. Mixed types are supported.

Supported types:

1. GlycoCT
2. IUPAC-condensed
3. IUPAC-extended
4. IUPAC-short
5. WURCS
6. Linear Code
7. pGlyco
8. StrucGP

Usage

auto_parse(x)

Arguments

x A character vector of structure strings.

Value

A `glyrepr::glycan_structure()` object.

Examples

```
# Single structure
x <- "Gal(b1-3)GlcNAc(b1-4)Glc(a1-" # IUPAC-condensed
auto_parse(x)

# Mixed types
x <- c(
  "Gal(b1-3)GlcNAc(b1-4)Glc(a1-", # IUPAC-condensed
  "Neu5Aca3Gal3(Fuca6)GlcNAcb-" # IUPAC-short
)
auto_parse(x)
```

parse_glycoct

Parse GlycoCT Structures

Description

This function parses GlycoCT strings into a `glyrepr::glycan_structure()`. GlycoCT is a format used by databases like GlyTouCan and GlyGen.

Usage

```
parse_glycoct(x)
```

Arguments

x A character vector of GlycoCT strings.

Details

GlycoCT format consists of two parts:

- RES: Contains monosaccharides (lines starting with 'b:') and substituents (lines starting with 's:')
- LIN: Contains linkage information between residues

For more information about GlycoCT format, see the `glycoct.md` documentation.

Value

A `glyrepr::glycan_structure()` object.

Examples

```
glycoct <- paste0(
  "RES\n",
  "1b:a-dgal-HEX-1:5\n",
  "2s:n-acetyl\n",
  "3b:b-dgal-HEX-1:5\n",
  "LIN\n",
  "1:1d(2+1)2n\n",
  "2:1o(3+1)3d"
)
parse_glycoct(glycoct)
```

parse_iupac_condensed *Parse IUPAC-condensed Structures*

Description

This function parses IUPAC-condensed strings into a `glyrepr:glycan_structure()`. For more information about IUPAC-condensed notation, see [doi:10.1351/pac199668101919](https://doi.org/10.1351/pac199668101919).

Usage

```
parse_iupac_condensed(x)
```

Arguments

x A character vector of IUPAC-condensed strings.

Details

The IUPAC-condensed notation is a compact form of IUPAC-extended notation. It is used by the **GlyConnect** database. It contains the following information:

- Monosaccharide name, e.g. "Gal", "GlcNAc", "Neu5Ac".
- Substituent, e.g. "9Ac", "4Ac", "3Me", "?S".
- Linkage, e.g. "b1-3", "a1-2", "a1-?".

An example of IUPAC-condensed string is "Gal(b1-3)GlcNAc(b1-4)Glc(a1-".

The reducing-end monosaccharide can be with or without anomer information. For example, the two strings below are all valid:

- "Neu5Ac(a2-"
- "Neu5Ac"

In the first case, the anomer is "a2". In the second case, the anomer is "?2".

Value

A `glyrepr::glycan_structure()` object.

See Also

`parse_iupac_short()`, `parse_iupac_extended()`

Examples

```
iupac <- "Gal(b1-3)GlcNAc(b1-4)Glc(a1-"  
parse_iupac_condensed(iupac)
```

parse_iupac_extended	<i>Parse IUPAC-extended Structures</i>
----------------------	--

Description

Parse IUPAC-extended-style structure characters into a `glyrepr::glycan_structure()`. For more information about IUPAC-extended format, see [doi:10.1351/pac199668101919](https://doi.org/10.1351/pac199668101919).

Usage

```
parse_iupac_extended(x)
```

Arguments

x	A character vector of IUPAC-extended strings.
---	---

Value

A `glyrepr::glycan_structure()` object.

See Also

`parse_iupac_condensed()`, `parse_iupac_short()`

Examples

```
iupac <- "\u03b2-D-Galp-(1\u21923)-\u03b2-D-GalpNAc-(1\u21922"  
parse_iupac_extended(iupac)
```

parse_iupac_short	<i>Parse IUPAC-short Structures</i>
-------------------	-------------------------------------

Description

Parse IUPAC-short-style structure characters into a `glyrepr::glycan_structure()`. For more information about IUPAC-short format, see [doi:10.1351/pac199668101919](https://doi.org/10.1351/pac199668101919).

Usage

```
parse_iupac_short(x)
```

Arguments

x	A character vector of IUPAC-short strings.
---	--

Details

The IUPAC-short notation is a compact form of IUPAC-condensed notation. It is rarely used in database, but appears a lot in literature for its conciseness. Compared with IUPAC-condensed notation, IUPAC-short notation ignore the anomer positions, assuming they are known for common monosaccharides. For example, "Neu5Aca3Gala-" assumes the anomer of Neu5Ac is C2 (a2-3 linked). Also, the parentheses around linkages are omitted, and parentheses are used to indicate branching, e.g. "Neu5Aca3Gala3(Fuca3)GlcNAcb-".

In the first case, the anomer is "a2". In the second case, the anomer is "?2".

Value

A `glyrepr::glycan_structure()` object.

See Also

`parse_iupac_condensed()`, `parse_iupac_extended()`

Examples

```
iupac <- "Neu5Aca3Gala3(Fuca6)GlcNAcb-"  
parse_iupac_short(iupac)
```

parse_linear_code	<i>Parse Linear Code Structures</i>
-------------------	-------------------------------------

Description

Parse Linear Code structures into a `glyrepr::glycan_structure()`. To know more about Linear Code, see [this article](#).

Usage

```
parse_linear_code(x)
```

Arguments

`x` A character vector of Linear Code strings.

Value

A `glyrepr::glycan_structure()` object.

Examples

```
linear_code <- "Ma3(Ma6)Mb4GNb4GNb"  
parse_linear_code(linear_code)
```

parse_pglyco_struc	<i>Parse pGlyco Structures</i>
--------------------	--------------------------------

Description

Parse pGlyco-style structure characters into a `glyrepr::glycan_structure()`. See example below for the structure format.

Usage

```
parse_pglyco_struc(x)
```

Arguments

`x` A character vector of pGlyco-style structure strings.

Value

A `glyrepr::glycan_structure()` object.

```
glycan <- parse_pglyco_struct("(N(F)(N(H(H(N))(H(N(H))))))")
print(glycan, verbose = TRUE)
```

parse_strucgp_struct	<i>Parse StrucGP Structures</i>
----------------------	---------------------------------

Parse StrucGP-style structure characters into a `glyrepr::glycan_structure()`. See example below for the structure format.

```
parse_structgp_struct(x)
```

x	A character vector of StrucGP-style structure strings.
---	--

A `glyrepr::glycan_structure()` object.

```
glycan <- parse_strucgp_struc("A2B2C1D1E2F1fedD1E2edcbB5ba")
print(glycan, verbose = TRUE)
```

parse_wurcs	<i>Parse WURCS Structures</i>
-------------	-------------------------------

This function parses WURCS strings into a `glyrepr:glycan_structure()`. Currently, only WURCS 2.0 is supported. For more information about WURCS, see [WURCS](#).

```
parse_wurcs(x)
```

x	A character vector of WURCS strings.
---	--------------------------------------

Value

A `glyrepr::glycan_structure()` object.

Examples

```
wurcs <- paste0(
  "WURCS=2.0/3,5,4/",
  "[a2122h-1b_1-5_2*NCC/3=0][a1122h-1b_1-5][a1122h-1a_1-5]/",
  "1-1-2-3-3/a4-b1_b4-c1_c3-d1_c6-e1"
)
parse_wurcs(wurcs)
```

Index

`auto_parse`, [2](#)

`glyrepr::glycan_structure()`, [2–9](#)

`parse_glycoct`, [3](#)

`parse_iupac_condensed`, [4](#)

`parse_iupac_condensed()`, [5, 6](#)

`parse_iupac_extended`, [5](#)

`parse_iupac_extended()`, [5, 6](#)

`parse_iupac_short`, [6](#)

`parse_iupac_short()`, [5](#)

`parse_linear_code`, [7](#)

`parse_pglyco_struc`, [7](#)

`parse_strucgp_struc`, [8](#)

`parse_wurcs`, [8](#)