Package 'glyparse'

August 28, 2026

Title Parsing Glycan Structure Text Representations

Version 0.8.1

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, IUPAC-compact, WURCS, LINUCS, Linear Code, GlycoCT, KCF, and GlycoWorkbench formats. It also provides an automatic parser to detect the format and parse the structure string.

License MIT + file LICENSE

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

Config/testthat/edition 3

Encoding UTF-8

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

Imports checkmate, cli, dplyr, glyrepr (>= 1.0.0), igraph, purrr (>= 1.0.0), rlang, rstackdeque, stringr

Depends R (>= 4.1)

VignetteBuilder knitr

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

RoxygenNote 7.3.3

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 2026-08-28 14:50:02 UTC

Contents

`auto_parse` *Automatic Structure Parsing*

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. GlyCAM IUPAC
6. IUPAC-compact
7. WURCS
8. Linear Code
9. pGlyco
10. StrucGP
11. KCF
12. LINUCS
13. GlycoWorkbench

Usage

```
auto_parse(x, on_failure = "error", progress = FALSE)
```

Arguments

x	A character vector of structure strings. NA values are allowed and will be returned as NA structures.
on_failure	How to handle parsing failures. "error" aborts when a structure cannot be parsed. "na" returns NA at invalid positions.
progress	Whether to show a progress bar while parsing.

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
  "Neu5Aca3Gala3(Fuca6)GlcNAcb-"  # IUPAC-short
)
auto_parse(x)
```

---

parse_glycam_iupac *Parse GlyCAM IUPAC Structures*

Description

Parse GlyCAM IUPAC-style structure strings into a glyrepr::glycan_structure().

Usage

```
parse_glycam_iupac(x, on_failure = "error", progress = FALSE)
```

Arguments

x	A character vector of GlyCAM IUPAC strings. NA values are allowed and will be returned as NA structures.
on_failure	How to handle parsing failures. "error" aborts when a structure cannot be parsed. "na" returns NA at invalid positions.
progress	Whether to show a progress bar while parsing.

Details

GlyCAM IUPAC is similar to IUPAC-condensed notation, but monosaccharides include configuration and ring markers such as `"DGlcp"` and `"LFucp"`, terminal reducing-end residues end in `"-OH"`, and residue modifiers are written in brackets, such as `"DGalp[6S]b1-4"`.

The parser normalizes GlyCAM IUPAC into IUPAC-condensed notation, then uses the IUPAC-condensed parser to construct the glycan structure. Explicit reducing-end moieties, such as `"-OH"` or `"-OME"`, are normalized to the regular reducing-end IUPAC-condensed form because glyrepr does not represent the terminal moiety separately.

Value

A `glyrepr::glycan_structure()` object.

See Also

`parse_iupac_condensed()`

Examples

```
glycam <- "DManpa1-3[DManpa1-6]DManpb1-4DGlcpNAcb1-OH"
parse_glycam_iupac(glycam)
```

`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, on_failure = "error", progress = FALSE, validate = TRUE)
```

Arguments

`x`	A character vector of GlycoCT strings. NA values are allowed and will be returned as NA structures.
`on_failure`	How to handle parsing failures. `"error"` aborts when a structure cannot be parsed. `"na"` returns `NA` at invalid positions.
`progress`	Whether to show a progress bar while parsing.
`validate`	Whether to validate parsed glycan graphs before constructing the result.

Details

GlycoCT format consists of:

- RES: Contains monosaccharides (lines starting with 'b:') and substituents (lines starting with 's:')
- LIN: Contains linkage information between residues
- UND: Contains floating substructures or substituents whose attachment to the main glycan is unresolved

Main reducing-end alditol residues retain their alditol status and use an unknown anomer configuration.

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_gwb *Parse GlycoWorkbench Structures*

Description

Parse GlycoWorkbench (GWB/GWS) structure strings into a `glyrepr::glycan_structure()`.

Usage

```
parse_gwb(x, on_failure = "error", progress = FALSE)
```

Arguments

x	A character vector of GlycoWorkbench strings. NA values are allowed and will be returned as NA structures.
on_failure	How to handle parsing failures. "error" aborts when a structure cannot be parsed. "na" returns NA at invalid positions.
progress	Whether to show a progress bar while parsing.

Details

GlycoWorkbench writes glycans from the reducing end towards the non-reducing ends. Residues include their anomer, configuration, and ring form, for example `"--4b1D-Gal,p"`. Branches are enclosed in parentheses, and the structure is followed by mass options after `$`.

The parser normalizes the glycan tree to IUPAC-condensed notation before constructing the glycan structure. GlycoWorkbench substituent nodes such as `"--6S"` and `"--9Ac"` are retained as monosaccharide substituents. Mass options are ignored because they are not part of the glycan graph. Explicit open-chain residues (`,o`) are supported only for a reduced `redEnd` root; other open-chain forms cannot be represented by `glyrepr`.

Value

A `glyrepr::glycan_structure()` object.

See Also

`parse_iupac_condensed()`

Examples

```
gwb <- paste0(
  "freeEnd--1b1D-GlcNAc,p(--6a1L-Fuc,p)",
  "--4b1D-Gal,p--3a2D-NeuAc,p$MONO,Und,0,0,freeEnd"
)
parse_gwb(gwb)
```

---

parse_iupac_compact *Parse IUPAC-compact Structures*

Description

Parse IUPAC-compact strings into a `glyrepr::glycan_structure()`.

Usage

```
parse_iupac_compact(x, on_failure = "error", progress = FALSE)
```

Arguments

`x`	A character vector of IUPAC-compact strings. NA values are allowed and will be returned as NA structures.
`on_failure`	How to handle parsing failures. `"error"` aborts when a structure cannot be parsed. `"na"` returns `NA` at invalid positions.
`progress`	Whether to show a progress bar while parsing.

Details

IUPAC-compact notation is similar to IUPAC-condensed notation, but linkages are written directly after the monosaccharide, such as `"Galb1-3GlcNAc"`, and branches are written in parentheses. The parser normalizes compact notation into IUPAC-condensed notation, then uses the IUPAC-condensed parser to construct the glycan structure.

Alditol glycans marked with `+aldi` retain their alditol status and use an unknown reducing-end anomer configuration.

Value

A `glyrepr::glycan_structure()` object.

See Also

`parse_iupac_condensed()`

Examples

```
iupac <- "Mana1-3(Mana1-6)Manb1-4GlcNAcb"
parse_iupac_compact(iupac)
```

---

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.

Usage

```
parse_iupac_condensed(x, on_failure = "error", progress = FALSE)
```

Arguments

`x`	A character vector of IUPAC-condensed strings. NA values are allowed and will be returned as NA structures.
`on_failure`	How to handle parsing failures. `"error"` aborts when a structure cannot be parsed. `"na"` returns `NA` at invalid positions.
`progress`	Whether to show a progress bar while parsing.

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 *Parse IUPAC-extended Structures*

Description

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

Usage

```
parse_iupac_extended(x, on_failure = "error", progress = FALSE)
```

Arguments

x	A character vector of IUPAC-extended strings. NA values are allowed and will be returned as NA structures.
on_failure	How to handle parsing failures. "error" aborts when a structure cannot be parsed. "na" returns NA at invalid positions.
progress	Whether to show a progress bar while parsing.

Details

The function accepts both a Unicode format (using the Greek letters alpha/beta and the arrow symbol ->) and a plain-text format (using the strings "alpha", "beta", and "->"). For example, both "\u03b2-D-Galp-(1\u21923)-\u03b1-D-GalpNAc-(1\u2192" and "beta-D-Galp-(1->3)-alpha-D-GalpNAc-(1->" are valid inputs.

Value

A glyrepr::glycan_structure() object.

See Also

parse_iupac_condensed(), parse_iupac_short()

Examples

```
iupac <- "\u03b2-D-Galp-(1\u21923)-\u03b1-D-GalpNAc-(1\u2192"
parse_iupac_extended(iupac)
parse_iupac_extended("beta-D-Galp-(1->3)-alpha-D-GalpNAc-(1->")
```

---

parse_iupac_short *Parse IUPAC-short Structures*

Description

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

Usage

```
parse_iupac_short(x, on_failure = "error", progress = FALSE)
```

Arguments

x	A character vector of IUPAC-short strings. NA values are allowed and will be returned as NA structures.
on_failure	How to handle parsing failures. "error" aborts when a structure cannot be parsed. "na" returns NA at invalid positions.
progress	Whether to show a progress bar while parsing.

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_kcf *Parse KCF Structures*

Description

This function parses KCF strings into a `glyrepr::glycan_structure()`. KCF is a graph-oriented format used by KEGG GLYCAN.

Usage

```
parse_kcf(x, on_failure = "error", progress = FALSE, validate = TRUE)
```

Arguments

x	A character vector of KCF strings. NA values are allowed and will be returned as NA structures.
on_failure	How to handle parsing failures. "error" aborts when a structure cannot be parsed. "na" returns NA at invalid positions.
progress	Whether to show a progress bar while parsing.
validate	Whether to validate parsed glycan graphs before constructing the result.

Value

A `glyrepr::glycan_structure()` object.

Examples

```
kcf <- paste0(
  "ENTRY       G00066                    Glycan\n",
  "NODE        6\n",
  "            1   Cer        18     0\n",
  "            2   Glc        12     0\n",
  "            3   Gal         6     0\n",
  "            4   GlcNAc     -2     0\n",
  "            5   Gal       -10     0\n",
  "            6   GlcNAc    -18     0\n",
  "EDGE        5\n",
  "            1     2:b1    1:1\n",
  "            2     3:b1    2:4\n",
  "            3     4:b1    3:3\n",
  "            4     5:b1    4:4\n",
  "            5     6:b1    5:3\n",
  "///"
)
parse_kcf(kcf)
```

---

parse_linear_code *Parse Linear Code Structures*

Description

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

Usage

```
parse_linear_code(x, on_failure = "error", progress = FALSE)
```

Arguments

`x`	A character vector of Linear Code strings. NA values are allowed and will be returned as NA structures.
`on_failure`	How to handle parsing failures. `"error"` aborts when a structure cannot be parsed. `"na"` returns `NA` at invalid positions.
`progress`	Whether to show a progress bar while parsing.

Value

A `glyrepr::glycan_structure()` object.

Examples

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

`parse_linucs` *Parse LINUCS Structures*

Description

Parse LINUCS strings into a `glyrepr::glycan_structure()`. LINUCS is a tree-oriented glycan format that writes each residue as a linkage token, a residue token, and a braced child list, for example `"[][Hexp]{[(4+1)][Hexp]{}}"`.

Usage

```
parse_linucs(x, on_failure = "error", progress = FALSE, validate = TRUE)
```

Arguments

`x`	A character vector of LINUCS strings. NA values are allowed and will be returned as NA structures.
`on_failure`	How to handle parsing failures. `"error"` aborts when a structure cannot be parsed. `"na"` returns `NA` at invalid positions.
`progress`	Whether to show a progress bar while parsing.
`validate`	Whether to validate parsed glycan graphs before constructing the result.

Details

LINUCS linkages are written as `"(parent+child)"`, where `parent` is the linkage position on the parent residue and `child` is the anomeric linkage position on the child residue. Residue labels are normalized to the monosaccharide and substituent vocabulary used by glyrepr::glyrepr. A `-ol` suffix on the root residue is retained as alditol status.

Value

A `glyrepr::glycan_structure()` object.

Examples

```
linucs <- "[][b-D-Glcp]{[(4+1)][b-D-Galp]{}}"
parse_linucs(linucs)
```

parse_pglyco_struc *Parse pGlyco Structures*

Description

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

Usage

```
parse_pglyco_struc(x, on_failure = "error", progress = FALSE, validate = TRUE)
```

Arguments

`x`	A character vector of pGlyco-style structure strings. NA values are allowed and will be returned as NA structures.
`on_failure`	How to handle parsing failures. `"error"` aborts when a structure cannot be parsed. `"na"` returns `NA` at invalid positions.
`progress`	Whether to show a progress bar while parsing.
`validate`	Whether to validate parsed glycan graphs before constructing the result.

Value

A `glyrepr::glycan_structure()` object.

Examples

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

parse_strucgp_struc *Parse StrucGP Structures*

Description

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

Usage

```
parse_strucgp_struc(x, on_failure = "error", progress = FALSE, validate = TRUE)
```

Arguments

x	A character vector of StrucGP-style structure strings. NA values are allowed and will be returned as NA structures.
on_failure	How to handle parsing failures. "error" aborts when a structure cannot be parsed. "na" returns NA at invalid positions.
progress	Whether to show a progress bar while parsing.
validate	Whether to validate parsed glycan graphs before constructing the result.

Value

A glyrepr::glycan_structure() object.

Examples

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

---

parse_wurcs *Parse WURCS Structures*

Description

This function parses WURCS strings into a glyrepr::glycan_structure(). Currently, only WURCS 2.0 is supported. For more information about WURCS, see WURCS. Main reducing-end alditol residues retain their alditol status and use an unknown anomer configuration. Ambiguous alternative linkage groups are represented as floating glycan parts when their child residue or subtree is not localized to one parent. Candidate parents may belong to the main glycan or another floating component. Floating substituents preserve their chemistry, carbon-position ambiguity, and candidate parent residues across the complete structure.

Usage

```
parse_wurcs(x, on_failure = "error", progress = FALSE, validate = TRUE)
```

Arguments

x	A character vector of WURCS strings. NA values are allowed and will be returned as NA structures.
on_failure	How to handle parsing failures. "error" aborts when a structure cannot be parsed. "na" returns NA at invalid positions.
progress	Whether to show a progress bar while parsing.
validate	Whether to validate parsed glycan graphs before constructing the result.

Value

A glyrepr::glycan_structure() object.

Examples

```
wurcs <- paste0(
  "WURCS=2.0/3,5,4/",
  "[a2122h-1b_1-5_2*NCC/3=O][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