

Package: glyrex (via r-universe)

October 18, 2026

Title Regular Expressions for Glycan Structures

Version 0.1.0

Description Compile and match glycan structural regular expressions with a stringr-like interface. Supports residue classes, linkage constraints, branches, repetition, assertions, and named captures on glyrepr graphs, including explicit reducing-end anomeric information.

License MIT + file LICENSE

Imports glyparse (>= 0.8.1), glyrepr (>= 1.1.0), igraph, utils

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

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

VignetteBuilder knitr

Config/pak/sysreqs libgmp-dev libicu-dev libxml2-dev

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

Date/Publication 2026-10-17 09:30:25 UTC

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

RemoteRef v0.1.0

RemoteSha bb9f7a54a95b81d2c1e234f64e6c4d417e7d7aa0

Contents

rex_compile	2
rex_count	3
rex_detect	4
rex_extract	5
rex_extract_all	6

rex_locate	7
rex_locate_all	8
rex_match	9
rex_match_all	10
rex_subset	11
rex_which	12
Index	13

rex_compile	<i>Compile a Glycan Regular Expression</i>
-------------	--

Description

Compile a reusable structural query. Patterns run from the non-reducing end towards the reducing end; they are not text regular expressions.

Usage

```
rex_compile(pattern)
```

Arguments

pattern One non-missing pattern string, or a compiled pattern.

Details

Compilation validates the pattern syntax and prepares it for reuse with query functions such as [rex_detect\(\)](#) and [rex_extract\(\)](#). Passing an already compiled pattern returns it unchanged.

Value

A glyrex_pattern containing a parsed syntax tree and capture names.

See Also

[vignette\("glyco-regex-syntax", package = "glyrex"\)](#) for a step-by-step guide to pattern syntax. See [rex_detect\(\)](#) for matching and result conventions.

Examples

```
p <- rex_compile("Galb4-(Fuca3-)GlcNAc")
rex_detect("Gal(b1-4)[Fuc(a1-3)]GlcNAc", p)
rex_detect(c("GlcNAc(a1-", "GlcNAc(b1-)", "GlcNAca$")
```

rex_count*Count Structural Matches in Glycans*

Description

Count the structural matches of a pattern in each glycan.

Usage

```
rex_count(glycans, pattern)
```

Arguments

glycans	A character vector of glycan expressions in any format supported by <code>glyparse::auto_parse()</code> , a <code>glyrepr_structure</code> vector, or one glyrepr-compatible <code>igraph</code> . Character inputs may mix formats; malformed structures throw an error.
pattern	A character vector of patterns or one <code>rex_compile()</code> object. Inputs recycle only from length one. Missing glycans or patterns propagate.

Details

The matching unit is a connected structural trace. As in `glycowork`, the first successful trace per starting node is retained, then strict subsets of other traces are removed. Distinct branches with equal sequences remain distinct matches. Matches are ordered by decreasing residue count and then starting node ID. Repeats are greedy unless marked lazy; successful traces may overlap. Counts therefore need not equal text-regex counts. Only connected rooted trees are supported; floating topology or floating substituents are rejected explicitly. See `rex_compile()` for syntax.

Value

An integer vector, with zero for no match and `NA_integer_` for missing glycans or patterns.

See Also

Other glycan queries: `rex_detect()`, `rex_extract()`, `rex_extract_all()`, `rex_locate()`, `rex_locate_all()`, `rex_match()`, `rex_match_all()`, `rex_subset()`, `rex_which()`

Examples

```
x <- c("Gal(b1-4)GlcNAc", "Fuc(a1-3)GlcNAc", NA)
rex_count(x, "HexNAc")
```

rex_detect

Detect Structural Matches in Glycans

Description

Test whether each glycan contains a structural pattern.

Usage

```
rex_detect(glycans, pattern, negate = FALSE)
```

Arguments

glycans	A character vector of glycan expressions in any format supported by <code>glyparse::auto_parse()</code> , a <code>glyrepr_structure</code> vector, or one glyrepr-compatible igraph. Character inputs may mix formats; malformed structures throw an error.
pattern	A character vector of patterns or one <code>rex_compile()</code> object. Inputs recycle only from length one. Missing glycans or patterns propagate.
negate	Invert detection when TRUE; missing values remain missing.

Details

The matching unit is a connected structural trace. As in glycowork, the first successful trace per starting node is retained, then strict subsets of other traces are removed. Distinct branches with equal sequences remain distinct matches. Matches are ordered by decreasing residue count and then starting node ID. Repeats are greedy unless marked lazy; successful traces may overlap. Counts therefore need not equal text-regex counts. Only connected rooted trees are supported; floating topology or floating substituents are rejected explicitly. See `rex_compile()` for syntax.

Value

A logical vector. Missing glycans or patterns return NA.

See Also

Other glycan queries: `rex_count()`, `rex_extract()`, `rex_extract_all()`, `rex_locate()`, `rex_locate_all()`, `rex_match()`, `rex_match_all()`, `rex_subset()`, `rex_which()`

Examples

```
x <- c("Gal(b1-4)GlcNAc", "Fuc(a1-3)GlcNAc", NA)
rex_detect(x, "Gal-HexNAc")
rex_detect(x, "Gal", negate = TRUE)
```

rex_extract

Extract the First Structural Match

Description

Extract the first matching fragment from each glycan.

Usage

```
rex_extract(glycans, pattern)
```

Arguments

glycans	A character vector of glycan expressions in any format supported by <code>glyparse::auto_parse()</code> , a <code>glyrepr_structure</code> vector, or one glyrepr-compatible <code>igraph</code> . Character inputs may mix formats; malformed structures throw an error.
pattern	A character vector of patterns or one <code>rex_compile()</code> object. Inputs recycle only from length one. Missing glycans or patterns propagate.

Details

The matching unit is a connected structural trace. As in glycowork, the first successful trace per starting node is retained, then strict subsets of other traces are removed. Distinct branches with equal sequences remain distinct matches. Matches are ordered by decreasing residue count and then starting node ID. Repeats are greedy unless marked lazy; successful traces may overlap. Counts therefore need not equal text-regex counts. Only connected rooted trees are supported; floating topology or floating substituents are rejected explicitly. See `rex_compile()` for syntax.

Value

A character vector of canonical glyrepr IUPAC strings for character input, or a glyrepr structure vector for structure or graph input. No match or missing input returns a missing value. A cut fragment retains the anomer and donor position of its former attachment linkage.

See Also

Other glycan queries: `rex_count()`, `rex_detect()`, `rex_extract_all()`, `rex_locate()`, `rex_locate_all()`, `rex_match()`, `rex_match_all()`, `rex_subset()`, `rex_which()`

Examples

```
x <- c("Gal(b1-4)GlcNAc", "Fuc(a1-3)GlcNAc", NA)
rex_extract(x, "HexNAc")
```

rex_extract_all	<i>Extract All Structural Matches</i>
-----------------	---------------------------------------

Description

Extract all matching fragments from each glycan.

Usage

```
rex_extract_all(glycans, pattern)
```

Arguments

glycans	A character vector of glycan expressions in any format supported by <code>glyparse::auto_parse()</code> , a <code>glyrepr_structure</code> vector, or one glyrepr-compatible <code>igraph</code> . Character inputs may mix formats; malformed structures throw an error.
pattern	A character vector of patterns or one <code>rex_compile()</code> object. Inputs recycle only from length one. Missing glycans or patterns propagate.

Details

The matching unit is a connected structural trace. As in `glycowork`, the first successful trace per starting node is retained, then strict subsets of other traces are removed. Distinct branches with equal sequences remain distinct matches. Matches are ordered by decreasing residue count and then starting node ID. Repeats are greedy unless marked lazy; successful traces may overlap. Counts therefore need not equal text-regex counts. Only connected rooted trees are supported; floating topology or floating substituents are rejected explicitly. See `rex_compile()` for syntax.

Value

A list of character vectors of canonical glyrepr IUPAC strings for character input, or glyrepr structure vectors for structure or graph input. No match returns an empty vector; missing input returns a length-one missing vector. A cut fragment retains the anomer and donor position of its former attachment linkage.

See Also

Other glycan queries: `rex_count()`, `rex_detect()`, `rex_extract()`, `rex_locate()`, `rex_locate_all()`, `rex_match()`, `rex_match_all()`, `rex_subset()`, `rex_which()`

Examples

```
x <- c("Gal(b1-4)GlcNAc", "Fuc(a1-3)GlcNAc", NA)
rex_extract_all(x, "HexNAc")
```

rex_locate	<i>Locate the First Structural Match</i>
------------	--

Description

Locate the nodes and edges of the first match in each glycan.

Usage

```
rex_locate(glycans, pattern)
```

Arguments

glycans	A character vector of glycan expressions in any format supported by <code>glyparse::auto_parse()</code> , a <code>glyrepr_structure</code> vector, or one glyrepr-compatible igraph. Character inputs may mix formats; malformed structures throw an error.
pattern	A character vector of patterns or one <code>rex_compile()</code> object. Inputs recycle only from length one. Missing glycans or patterns propagate.

Details

The matching unit is a connected structural trace. As in glycowork, the first successful trace per starting node is retained, then strict subsets of other traces are removed. Distinct branches with equal sequences remain distinct matches. Matches are ordered by decreasing residue count and then starting node ID. Repeats are greedy unless marked lazy; successful traces may overlap. Counts therefore need not equal text-regex counts. Only connected rooted trees are supported; floating topology or floating substituents are rejected explicitly. See `rex_compile()` for syntax.

Value

A data frame with integer list-columns `nodes` and `edges`, and one row per input. No match or missing input uses `NA_integer_` in both columns. IDs are one-based and refer to the input graph, or the canonical graph obtained from character input, as in `glyrepr::structure_nodes()` and `glyrepr::structure_edges()`. Only edges within the match are returned.

See Also

Other glycan queries: `rex_count()`, `rex_detect()`, `rex_extract()`, `rex_extract_all()`, `rex_locate_all()`, `rex_match()`, `rex_match_all()`, `rex_subset()`, `rex_which()`

Examples

```
x <- c("Gal(b1-4)GlcNAc", "Fuc(a1-3)GlcNAc", NA)
rex_locate(x, "Gal-GlcNAc")
```

rex_locate_all	<i>Locate All Structural Matches</i>
----------------	--------------------------------------

Description

Locate the nodes and edges of all matches in each glycan.

Usage

```
rex_locate_all(glycans, pattern)
```

Arguments

glycans	A character vector of glycan expressions in any format supported by <code>glyparse::auto_parse()</code> , a <code>glyrepr_structure</code> vector, or one glyrepr-compatible igraph. Character inputs may mix formats; malformed structures throw an error.
pattern	A character vector of patterns or one <code>rex_compile()</code> object. Inputs recycle only from length one. Missing glycans or patterns propagate.

Details

The matching unit is a connected structural trace. As in glycowork, the first successful trace per starting node is retained, then strict subsets of other traces are removed. Distinct branches with equal sequences remain distinct matches. Matches are ordered by decreasing residue count and then starting node ID. Repeats are greedy unless marked lazy; successful traces may overlap. Counts therefore need not equal text-regex counts. Only connected rooted trees are supported; floating topology or floating substituents are rejected explicitly. See `rex_compile()` for syntax.

Value

A list of data frames with integer list-columns nodes and edges. No match produces zero rows; missing input produces one row with NA_integer_ in both columns. IDs are one-based and refer to the input graph, or the canonical graph obtained from character input, as in `glyrepr::structure_nodes()` and `glyrepr::structure_edges()`. Only edges within the match are returned.

See Also

Other glycan queries: `rex_count()`, `rex_detect()`, `rex_extract()`, `rex_extract_all()`, `rex_locate()`, `rex_match()`, `rex_match_all()`, `rex_subset()`, `rex_which()`

Examples

```
x <- c("Gal(b1-4)GlcNAc", "Fuc(a1-3)GlcNAc", NA)
rex_locate_all(x, "Gal-GlcNAc")
```

rex_match

*Extract the First Structural Match and Named Captures***Description**

Extract the first match and its named capture groups from each glycan.

Usage

```
rex_match(glycans, pattern)
```

Arguments

glycans	A character vector of glycan expressions in any format supported by glyparse::auto_parse() , a glyrepr_structure vector, or one glyrepr-compatible igraph. Character inputs may mix formats; malformed structures throw an error.
pattern	A character vector of patterns or one rex_compile() object. Inputs recycle only from length one. Missing glycans or patterns propagate.

Details

The matching unit is a connected structural trace. As in glycowork, the first successful trace per starting node is retained, then strict subsets of other traces are removed. Distinct branches with equal sequences remain distinct matches. Matches are ordered by decreasing residue count and then starting node ID. Repeats are greedy unless marked lazy; successful traces may overlap. Counts therefore need not equal text-regex counts. Only connected rooted trees are supported; floating topology or floating substituents are rejected explicitly. See [rex_compile\(\)](#) for syntax.

Value

A character matrix with one row per input, a match column, and columns for named captures. No match or missing input produces a missing row; unmatched captures are NA. Fragments are canonical glyrepr IUPAC strings. Vectorized patterns must have identical capture names.

See Also

Other glycan queries: [rex_count\(\)](#), [rex_detect\(\)](#), [rex_extract\(\)](#), [rex_extract_all\(\)](#), [rex_locate\(\)](#), [rex_locate_all\(\)](#), [rex_match_all\(\)](#), [rex_subset\(\)](#), [rex_which\(\)](#)

Examples

```
x <- c("Gal(b1-4)GlcNAc", "Fuc(a1-3)GlcNAc", NA)
rex_match(x, "(?<terminal>Gal)-GlcNAc")
```

 rex_match_all

Extract All Structural Matches and Named Captures

Description

Extract all matches and their named capture groups from each glycan.

Usage

```
rex_match_all(glycans, pattern)
```

Arguments

glycans	A character vector of glycan expressions in any format supported by <code>glyparse::auto_parse()</code> , a <code>glyrepr_structure</code> vector, or one glyrepr-compatible igraph. Character inputs may mix formats; malformed structures throw an error.
pattern	A character vector of patterns or one <code>rex_compile()</code> object. Inputs recycle only from length one. Missing glycans or patterns propagate.

Details

The matching unit is a connected structural trace. As in glycowork, the first successful trace per starting node is retained, then strict subsets of other traces are removed. Distinct branches with equal sequences remain distinct matches. Matches are ordered by decreasing residue count and then starting node ID. Repeats are greedy unless marked lazy; successful traces may overlap. Counts therefore need not equal text-regex counts. Only connected rooted trees are supported; floating topology or floating substituents are rejected explicitly. See `rex_compile()` for syntax.

Value

A list of character matrices with a match column followed by named capture columns. No match produces zero rows; missing input produces one missing row. Unmatched captures are NA. Fragments are canonical glyrepr IUPAC strings. Vectorized patterns must have identical capture names.

See Also

Other glycan queries: `rex_count()`, `rex_detect()`, `rex_extract()`, `rex_extract_all()`, `rex_locate()`, `rex_locate_all()`, `rex_match()`, `rex_subset()`, `rex_which()`

Examples

```
x <- c("Gal(b1-4)GlcNAc", "Fuc(a1-3)GlcNAc", NA)
rex_match_all(x, "(?<terminal>Gal)-GlcNAc")
```

rex_subset

*Subset Glycans by Structural Matches***Description**

Keep glycans that match a structural pattern.

Usage

```
rex_subset(glycans, pattern, negate = FALSE)
```

Arguments

glycans	A character vector of glycan expressions in any format supported by <code>glyparse::auto_parse()</code> , a <code>glyrepr_structure</code> vector, or one glyrepr-compatible <code>igraph</code> . Character inputs may mix formats; malformed structures throw an error.
pattern	A character vector of patterns or one <code>rex_compile()</code> object. Inputs recycle only from length one. Missing glycans or patterns propagate.
negate	Invert detection when TRUE; missing values remain missing.

Details

The matching unit is a connected structural trace. As in glycowork, the first successful trace per starting node is retained, then strict subsets of other traces are removed. Distinct branches with equal sequences remain distinct matches. Matches are ordered by decreasing residue count and then starting node ID. Repeats are greedy unless marked lazy; successful traces may overlap. Counts therefore need not equal text-regex counts. Only connected rooted trees are supported; floating topology or floating substituents are rejected explicitly. See `rex_compile()` for syntax.

Value

A subset preserving the input representation and names. Missing results are omitted. For a single graph input, returns the graph on a match and NULL otherwise. The pattern must have length one or the length of glycans.

See Also

Other glycan queries: `rex_count()`, `rex_detect()`, `rex_extract()`, `rex_extract_all()`, `rex_locate()`, `rex_locate_all()`, `rex_match()`, `rex_match_all()`, `rex_which()`

Examples

```
x <- c("Gal(b1-4)GlcNAc", "Fuc(a1-3)GlcNAc", NA)
rex_subset(x, "Gal")
rex_subset(x, "Gal", negate = TRUE)
```

rex_which

*Find Indices of Glycans with Structural Matches***Description**

Return the indices of glycans that match a structural pattern.

Usage

```
rex_which(glycans, pattern, negate = FALSE)
```

Arguments

glycans	A character vector of glycan expressions in any format supported by <code>glyparse::auto_parse()</code> , a <code>glyrepr_structure</code> vector, or one glyrepr-compatible igraph. Character inputs may mix formats; malformed structures throw an error.
pattern	A character vector of patterns or one <code>rex_compile()</code> object. Inputs recycle only from length one. Missing glycans or patterns propagate.
negate	Invert detection when TRUE; missing values remain missing.

Details

The matching unit is a connected structural trace. As in glycowork, the first successful trace per starting node is retained, then strict subsets of other traces are removed. Distinct branches with equal sequences remain distinct matches. Matches are ordered by decreasing residue count and then starting node ID. Repeats are greedy unless marked lazy; successful traces may overlap. Counts therefore need not equal text-regex counts. Only connected rooted trees are supported; floating topology or floating substituents are rejected explicitly. See `rex_compile()` for syntax.

Value

An integer vector of matching input indices. Missing results are omitted. The pattern must have length one or the length of glycans.

See Also

Other glycan queries: `rex_count()`, `rex_detect()`, `rex_extract()`, `rex_extract_all()`, `rex_locate()`, `rex_locate_all()`, `rex_match()`, `rex_match_all()`, `rex_subset()`

Examples

```
x <- c("Gal(b1-4)GlcNAc", "Fuc(a1-3)GlcNAc", NA)
rex_which(x, "Fuc")
rex_which(x, "Gal", negate = TRUE)
```

Index

* glycan queries

- rex_count, [3](#)
- rex_detect, [4](#)
- rex_extract, [5](#)
- rex_extract_all, [6](#)
- rex_locate, [7](#)
- rex_locate_all, [8](#)
- rex_match, [9](#)
- rex_match_all, [10](#)
- rex_subset, [11](#)
- rex_which, [12](#)

glyparse::auto_parse(), [3–12](#)
glyrepr::structure_edges(), [7, 8](#)
glyrepr::structure_nodes(), [7, 8](#)

rex_compile, [2](#)
rex_compile(), [3–12](#)
rex_count, [3, 4–12](#)
rex_detect, [3, 4, 5–12](#)
rex_detect(), [2](#)
rex_extract, [3, 4, 5, 6–12](#)
rex_extract(), [2](#)
rex_extract_all, [3–5, 6, 7–12](#)
rex_locate, [3–6, 7, 8–12](#)
rex_locate_all, [3–7, 8, 9–12](#)
rex_match, [3–8, 9, 10–12](#)
rex_match_all, [3–9, 10, 11, 12](#)
rex_subset, [3–10, 11, 12](#)
rex_which, [3–11, 12](#)