Package: glyread (via r-universe)

May 14, 2026

Title Read and process glycomics and glycoproteomics data

Version 0.11.0

Description Provides a unified interface for reading and processing quantification results from various glycomics and glycoproteomics software tools, including Byonic (ByoLogic, pGlycoQuant), StrucGP, pGlyco3 (pGlycoQuant), Glyco-Decipher, GlyHunter, and MSFragger. The package standardizes raw data from different sources, performs peptide-spectrum match (PSM) aggregation, parses glycan compositions and structures, and converts the data into a standardized experiment object from the 'glyexp' package for downstream analysis within the glycoverse ecosystem.

License MIT + file LICENSE

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

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://glycoverse.github.io/glyread/

Imports AnnotationDbi, checkmate, cli, dplyr, fs, glyexp (>= 0.12.1), glyparse (>= 0.5.3), glyrepr (>= 0.7.4), janitor, magrittr, purrr, readr, readxl, rlang, stringr, tibble, tidyr, tidyselect

Depends R (>= 4.1)

VignetteBuilder knitr

Config/pak/sysreqs cmake libglpk-dev make libicu-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libx11-dev zlib1g-dev

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

Date/Publication 2026-05-02 14:41:49 UTC

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

RemoteRef v0.11.0

RemoteSha d2d8d25f71c8e6ad8742e5e8b3bed5c24595dcf2

Contents

`read_byonic_byologic` *Read Byonic-Byologic result*

Description

If you used Byonic for intact glycopeptide identification, and used Byologic for quantification, this is the function for you. It reads in a result file and returns a `glyexp::experiment()` object. Currently only label-free quantification is supported.

Usage

```
read_byonic_byologic(
  fp,
  sample_info = NULL,
  quant_method = "label-free",
  glycan_type = "N",
  sample_name_converter = NULL,
  orgdb = "org.Hs.eg.db",
  multisite = "expand"
)
```

Arguments

`fp`	File path of the pGlycoQuant result file.
`sample_info`	File path of the sample information file (csv), or a sample information data.frame/tibble.
`quant_method`	Quantification method. Either "label-free" or "TMT".
`glycan_type`	Glycan type. One of "N", "O-GalNAc", "O-GlcNAc", "O-Man", "O-Fuc", or "O-Glc". Default is "N".
`sample_name_converter`	A function to convert sample names from file paths. The function should take a character vector of old sample names and return new sample names. Note that sample names in `sample_info` should match the new names. If NULL, original names are kept.

`orgdb` name of the OrgDb package to use for UniProt to gene symbol conversion. Default is "org.Hs.eg.db".

`multisite` How to handle multisite glycopeptides.

- "expand" (default) expands each multisite glycopeptide into site-specific rows.
- "drop" removes multisite glycopeptides.

Value

An `glyexp::experiment()` object.

Which file to use?

Open the .blgc file in the result folder with PMI-Byos. In the "Peptide List" panel (usually on the bottom right), click "Export content of the table to a CSV file" button. The exported .csv file is the file you should use.

Multisite glycopeptides

Some glycopeptides can have more than one glycosylation site. By default, they are expanded into multiple rows with the same quantification value but different `protein_site` and `glycan_composition`. Set `multisite = "drop"` to remove multisite glycopeptides instead.

Variable information

The following columns could be found in the variable information tibble:

- `peptide`: character, peptide sequence
- `peptide_site`: integer, site of glycosylation on peptide
- `protein`: character, protein accession
- `protein_site`: integer, site of glycosylation on protein
- `gene`: character, gene name (symbol)
- `glycan_composition`: `glyrepr::glycan_composition()`, glycan compositions.

Sample information

The sample information file should be a `csv` file with the first column named `sample`, and the rest of the columns being sample information. The `sample` column must match the `RawName` column in the pGlyco3 result file, although the order can be different.

You can put any useful information in the sample information file. Recommended columns are:

- `group`: grouping or conditions, e.g. "control" or "tumor", required for most downstream analyses
- `batch`: batch information, required for batch effect correction

Aggregation

pGlyco3 performs quantification on the PSM level. This level of information is too detailed for most downstream analyses. This function aggregate PSMs into glycopeptides through summation. For each glycopeptide (unique combination of "peptide", "peptide_site", "protein", "protein_site", "gene", "glycan_composition", "glycan_structure"), we sum up the quantifications of all PSMs that belong to this glycopeptide.

See Also

`glyexp::experiment()`, `glyrepr::glycan_composition()`

---

read_byonic_pglycoquant

Read Byonic-pGlycoQuant result

Description

If you used Byonic for intact glycopeptide identification, and used pGlycoQuant for quantification, this is the function for you. It reads in a pGlycoQuant result file and returns a `glyexp::experiment()` object. Currently only label-free quantification is supported.

Usage

```
read_byonic_pglycoquant(
  fp,
  sample_info = NULL,
  quant_method = "label-free",
  glycan_type = "N",
  sample_name_converter = NULL,
  orgdb = "org.Hs.eg.db",
  parse_structure = TRUE,
  multisite = "expand"
)
```

Arguments

`fp`	File path of the pGlycoQuant result file.
`sample_info`	File path of the sample information file (csv), or a sample information data.frame/tibble.
`quant_method`	Quantification method. Either "label-free" or "TMT".
`glycan_type`	Glycan type. One of "N", "O-GalNAc", "O-GlcNAc", "O-Man", "O-Fuc", or "O-Glc". Default is "N".
`sample_name_converter`	A function to convert sample names from file paths. The function should take a character vector of old sample names and return new sample names. Note that sample names in `sample_info` should match the new names. If NULL, original names are kept.

`orgdb` name of the OrgDb package to use for UniProt to gene symbol conversion. Default is "org.Hs.eg.db".

`parse_structure`
Logical. Whether to parse glycan structures. If `TRUE`, glycan structures are parsed and included in the `var_info` as `glycan_structure` column. If `FALSE` (default), structure parsing is skipped and structure-related columns are removed.

`multisite` How to handle multisite glycopeptides.

- "expand" (default) expands each multisite glycopeptide into site-specific rows.
- "drop" removes multisite glycopeptides.

Value

An `glyexp::experiment()` object.

Which file to use?

You should use the "Quant.spectra.list" file in the pGlycoQuant result folder. Files from Byonic result folder are not needed. For instructions on how to use Byonic and pGlycoQuant, please refer to the manual: pGlycoQuant.

Multisite glycopeptides

Some glycopeptides can have more than one glycosylation site. By default, they are expanded into multiple rows with the same quantification value but different `protein_site` and `glycan_composition`. Set `multisite = "drop"` to remove multisite glycopeptides instead.

Variable information

The following columns could be found in the variable information tibble:

- `peptide`: character, peptide sequence
- `peptide_site`: integer, site of glycosylation on peptide
- `protein`: character, protein accession
- `protein_site`: integer, site of glycosylation on protein
- `gene`: character, gene name (symbol)
- `glycan_composition`: `glyrepr::glycan_composition()`, glycan compositions.
- `glycan_structure`: `glyrepr::glycan_structure()`, glycan structures (if `parse_structure = TRUE`).

Sample information

The sample information file should be a `csv` file with the first column named `sample`, and the rest of the columns being sample information. The `sample` column must match the `RawName` column in the pGlyco3 result file, although the order can be different.

You can put any useful information in the sample information file. Recommended columns are:

- `group`: grouping or conditions, e.g. "control" or "tumor", required for most downstream analyses
- `batch`: batch information, required for batch effect correction

Aggregation

pGlyco3 performs quantification on the PSM level. This level of information is too detailed for most downstream analyses. This function aggregate PSMs into glycopeptides through summation. For each glycopeptide (unique combination of "peptide", "peptide_site", "protein", "protein_site", "gene", "glycan_composition", "glycan_structure"), we sum up the quantifications of all PSMs that belong to this glycopeptide.

See Also

`glyexp::experiment()`, `glyrepr::glycan_composition()`

`read_glycan_finder` *Read GlycanFinder result*

Description

GlycanFinder is a software for intact glycopeptide identification. This function reads in the result file and returns a `glyexp::experiment()` object. Currently only label-free quantification is supported.

Usage

```
read_glycan_finder(
  fp,
  sample_info = NULL,
  quant_method = "label-free",
  glycan_type = "N",
  sample_name_converter = NULL,
  orgdb = "org.Hs.eg.db",
  parse_structure = TRUE
)
```

Arguments

`fp`	File path of the GlycanFinder result file.
`sample_info`	File path of the sample information file (csv), or a sample information data.frame/tibble.
`quant_method`	Quantification method. Either "label-free" or "TMT".
`glycan_type`	Glycan type. One of "N", "O-GalNAc", "O-GlcNAc", "O-Man", "O-Fuc", or "O-Glc". Default is "N".

`sample_name_converter`
: A function to convert sample names from file paths. The function should take a character vector of old sample names and return new sample names. Note that sample names in `sample_info` should match the new names. If NULL, original names are kept.

`orgdb`
: Name of the OrgDb package to use for UniProt to gene symbol conversion. Default is "org.Hs.eg.db".

`parse_structure`
: Logical. Whether to parse glycan structures. If `TRUE` (default), glycan structures are parsed and included in the `var_info` as `glycan_structure` column. If `FALSE`, structure parsing is skipped and structure-related columns are removed.

Value

An `glyexp::experiment()` object.

Which file to use?

You should use the `lfq/lfq.protein-glycopeptides.csv` file from the GlycanFinder result folder. This file contains the quantification information for glycopeptides.

Sample information

The sample information file should be a `csv` file with the first column named `sample`, and the rest of the columns being sample information. The `sample` column must match the sample names in the Area columns of the GlycanFinder result file (e.g., "C_3", "H_3"), although the order can be different.

Variable information

The following columns could be found in the variable information tibble:

- `peptide`: character, peptide sequence
- `peptide_site`: integer, site of glycosylation on peptide
- `protein`: character, protein accession
- `protein_site`: integer, site of glycosylation on protein
- `gene`: character, gene name (symbol)
- `glycan_composition`: `glyrepr::glycan_composition()`, glycan compositions.
- `glycan_structure`: `glyrepr::glycan_structure()`, glycan structures (if `parse_structure = TRUE`).

See Also

`glyexp::experiment()`, `glyrepr::glycan_composition()`, `glyrepr::glycan_structure()`

`read_glyco_decipher` *Read glyco-decipher output*

Description

Glyco-Decipher is a software for glycopeptide identification and quantification. This function reads in the result file and returns a `glyexp::experiment()` object. Currently only label-free quantification is supported.

Usage

```
read_glyco_decipher(
  fp,
  sample_info = NULL,
  quant_method = "label-free",
  glycan_type = "N",
  sample_name_converter = NULL,
  orgdb = "org.Hs.eg.db"
)
```

Arguments

`fp`	File path of the pGlyco3 result file.
`sample_info`	File path of the sample information file (csv), or a sample information data.frame/tibble.
`quant_method`	Quantification method. Either "label-free" or "TMT".
`glycan_type`	Glycan type. One of "N", "O-GalNAc", "O-GlcNAc", "O-Man", "O-Fuc", or "O-Glc". Default is "N".
`sample_name_converter`	A function to convert sample names from file paths. The function should take a character vector of old sample names and return new sample names. Note that sample names in `sample_info` should match the new names. If NULL, original names are kept.
`orgdb`	name of the OrgDb package to use for UniProt to gene symbol conversion. Default is "org.Hs.eg.db".

Value

An `glyexp::experiment()` object.

Which file to use?

You should use the "site.csv" file in the result folder. This file contains "Site" and "Glycan" columns, followed by quantification result for each sample.

Protein inference and uncertain sites

Glyco-Decipher reports uncertain sites and proteins. This function automatically performs protein inference using the parsimony method to find the leader proteins. This ensures each glycopeptide is uniquely mapped to a single protein, gene, and glycosite. Besides, uncertain sites on proteins are assigned `NA`.

Variable information

The following columns could be found in the variable information tibble:

- `protein`: character, protein accession (after protein inference)
- `protein_site`: integer, site of glycosylation on protein (after protein inference)
- `gene`: character, gene name (symbol) (after protein inference)
- `glycan_composition`: `glyrepr::glycan_composition()`, glycan compositions.

Sample information

The sample information file should be a `csv` file with the first column named `sample`, and the rest of the columns being sample information. The `sample` column must match the `RawName` column in the pGlyco3 result file, although the order can be different.

You can put any useful information in the sample information file. Recommended columns are:

- `group`: grouping or conditions, e.g. "control" or "tumor", required for most downstream analyses
- `batch`: batch information, required for batch effect correction

---

`read_glyhunter` *Read GlyHunter result*

Description

This function reads in a GlyHunter result file and returns a `glyexp::experiment()` object.

Usage

```
read_glyhunter(
  fp,
  sample_info = NULL,
  preset = c("Fu_NC_2026", "Fu_NP_2026"),
  glycan_type = "N",
  sample_name_converter = NULL
)
```

Arguments

`fp`	File path of the GlyHunter result file.
`sample_info`	File path of the sample information file (csv), or a sample information data.frame/tibble.
`preset`	"Fu_NC_2026" (default) or "Fu_NP_2026". Use "Fu_NC_2026" to use the configuration following "DOI: 10.1038/s41467-026-68579-x". Use "Fu_NP_2026" to use the configuration in an unpublished Nature Protocols paper.
`glycan_type`	Glycan type. One of "N", "O-GalNAc", "O-GlcNAc", "O-Man", "O-Fuc", or "O-Glc". Default is "N".
`sample_name_converter`	A function to convert sample names from file paths. The function should take a character vector of old sample names and return new sample names. Note that sample names in `sample_info` should match the new names. If NULL, original names are kept.

Value

An `glyexp::experiment()` object.

Which file to use?

Use the "summary_area.csv" file in the GlyHunter result folder. No edit is necessary.

Variable information

The following columns could be found in the variable information tibble:

- `glycan_composition`: `glyrepr::glycan_composition()`, glycan compositions.

If "Fu_NP_2026" preset is used, two additional columns will be included:

- `nL`: Number of 2,3-linked sialic acids.
- `nE`: Number of 2,6-linked sialic acids.

Sample information

The sample information file should be a `csv` file with the first column named `sample`, and the rest of the columns being sample information. The `sample` column must match the `RawName` column in the pGlyco3 result file, although the order can be different.

You can put any useful information in the sample information file. Recommended columns are:

- `group`: grouping or conditions, e.g. "control" or "tumor", required for most downstream analyses
- `batch`: batch information, required for batch effect correction

See Also

`glyexp::experiment()`, `glyrepr::glycan_composition()`

`read_msfragger` *Read MSFragger-Glyco result*

Description

MSFragger-Glyco is a software for glycopeptide identification and quantification. This function reads in the result file and returns a `glyexp::experiment()` object.

Usage

```
read_msfragger(
  dp,
  sample_info = NULL,
  quant_method = "label-free",
  glycan_type = "N",
  sample_name_converter = NULL
)
```

Arguments

`dp`	The directory path of the MSFragger-Glyco result folder.
`sample_info`	File path of the sample information file (csv), or a sample information data.frame/tibble.
`quant_method`	Quantification method. Either "label-free" or "TMT".
`glycan_type`	Glycan type. One of "N", "O-GalNAc", "O-GlcNAc", "O-Man", "O-Fuc", or "O-Glc". Default is "N".
`sample_name_converter`	A function to convert sample names from file paths. The function should take a character vector of old sample names and return new sample names. Note that sample names in `sample_info` should match the new names. If NULL, original names are kept.

Details

This function uses the "psm.tsv" file in each sample folder. Sample names are extracted from the file paths. They are the parent of each "psm.tsv" file. For example, "msfragger_result/H1/psm.tsv" will be named "H1".

Value

An `glyexp::experiment()` object.

Variable information

The following columns could be found in the variable information tibble:

- `peptide`: character, peptide sequence

- `peptide_site`: integer, site of glycosylation on peptide
- `protein`: character, protein accession
- `protein_site`: integer, site of glycosylation on protein
- `gene`: character, gene name (symbol)
- `glycan_composition`: `glyrepr::glycan_composition()`, glycan compositions.

See Also

`glyexp::experiment()`, `glyrepr::glycan_composition()`

`read_pglyco3` *Read pGlyco3 result*

Description

pGlyco3 is a software for intact glycopeptide identification and quantification. This function reads in the result file and returns a `glyexp::experiment()` object. Currently only label-free quantification is supported.

Use this function if you only use pGlyco3. If you also use pGlycoQuant for quantification, use `read_pglyco3_pglycoquant()` instead.

Usage

```
read_pglyco3(
  fp,
  sample_info = NULL,
  quant_method = "label-free",
  glycan_type = "N",
  sample_name_converter = NULL,
  parse_structure = FALSE
)
```

Arguments

`fp`	File path of the pGlyco3 result file.
`sample_info`	File path of the sample information file (csv), or a sample information data.frame/tibble.
`quant_method`	Quantification method. Either "label-free" or "TMT".
`glycan_type`	Glycan type. One of "N", "O-GalNAc", "O-GlcNAc", "O-Man", "O-Fuc", or "O-Glc". Default is "N".
`sample_name_converter`	A function to convert sample names from file paths. The function should take a character vector of old sample names and return new sample names. Note that sample names in `sample_info` should match the new names. If NULL, original names are kept.

`parse_structure`
: Logical. Whether to parse glycan structures. If `TRUE`, glycan structures are parsed and included in the `var_info` as `glycan_structure` column. If `FALSE` (default), structure parsing is skipped and structure-related columns are removed.

Value

An `glyexp::experiment()` object.

Which file to use?

You should use the file with "-Pro-Quant" suffix that contains quantification information. The file should have columns including `RawName`, `MonoArea`, `Peptide`, `Proteins`, `Genes`, `GlycanComposition`, `PlausibleStruct`, `GlySite`, and `ProSites`.

Sample information

The sample information file should be a `csv` file with the first column named `sample`, and the rest of the columns being sample information. The `sample` column must match the `RawName` column in the pGlyco3 result file, although the order can be different.

You can put any useful information in the sample information file. Recommended columns are:

- `group`: grouping or conditions, e.g. "control" or "tumor", required for most downstream analyses
- `batch`: batch information, required for batch effect correction

Protein inference

pGlyco3 reports protein groups. That is, shared glycopeptides are reported as a group of proteins separated by ";". This function automatically performs protein inference using the parsimony method to find the leader proteins. This ensures each glycopeptide is uniquely mapped to a single protein, gene, and glycosite.

Aggregation

pGlyco3 performs quantification on the PSM level. This level of information is too detailed for most downstream analyses. This function aggregate PSMs into glycopeptides through summation. For each glycopeptide (unique combination of "peptide", "peptide_site", "protein", "protein_site", "gene", "glycan_composition", "glycan_structure"), we sum up the quantifications of all PSMs that belong to this glycopeptide.

Variable information

The following columns could be found in the variable information tibble:

- `peptide`: character, peptide sequence
- `peptide_site`: integer, site of glycosylation on peptide
- `protein`: character, protein accession (after protein inference)

- `protein_site`: integer, site of glycosylation on protein (after protein inference)
- `gene`: character, gene name (symbol) (after protein inference)
- `glycan_composition`: `glyrepr::glycan_composition()`, glycan compositions.
- `glycan_structure`: `glyrepr::glycan_structure()`, glycan structures (if `parse_structure = TRUE`).

Glycan structures

pGlyco3 reports a "plausible structure" for each glycan. You can set `parse_structure = TRUE` to parse these structures into a "glycan_structure" column as a `glyrepr::glycan_structure()` vector. However, please take caution with these structures, because pGlyco3 does not have strict quality control on glycan structure annotations.

See Also

`glyexp::experiment()`, `glyrepr::glycan_composition()`, `glyrepr::glycan_structure()`

---

read_pglyco3_pglycoquant

Read pGlyco3-pGlycoQuant result

Description

If you used pGlyco3 for intact glycopeptide identification, and used pGlycoQuant for quantification, this is the function for you. It reads in a pGlycoQuant result file and returns a `glyexp::experiment()` object. Currently only label-free quantification is supported.

Usage

```
read_pglyco3_pglycoquant(
  fp,
  sample_info = NULL,
  quant_method = "label-free",
  glycan_type = "N",
  sample_name_converter = NULL,
  parse_structure = FALSE
)
```

Arguments

`fp`	File path of the pGlycoQuant result file.
`sample_info`	File path of the sample information file (csv), or a sample information data.frame/tibble.
`quant_method`	Quantification method. Either "label-free" or "TMT".
`glycan_type`	Glycan type. One of "N", "O-GalNAc", "O-GlcNAc", "O-Man", "O-Fuc", or "O-Glc". Default is "N".

`sample_name_converter`
: A function to convert sample names from file paths. The function should take a character vector of old sample names and return new sample names. Note that sample names in `sample_info` should match the new names. If NULL, original names are kept.

`parse_structure`
: Logical. Whether to parse glycan structures. If `TRUE`, glycan structures are parsed and included in the `var_info` as `glycan_structure` column. If `FALSE` (default), structure parsing is skipped and structure-related columns are removed.

Value

An `glyexp::experiment()` object.

Which file to use?

You should use the "Quant.spectra.list" file in the pGlycoQuant result folder. Files from pGlyco3 result folder are not needed. For instructions on how to use pGlyco3 and pGlycoQuant, please refer to the manual: pGlycoQuant.

Variable information

The following columns could be found in the variable information tibble:

- `peptide`: character, peptide sequence
- `peptide_site`: integer, site of glycosylation on peptide
- `protein`: character, protein accession (after protein inference)
- `protein_site`: integer, site of glycosylation on protein (after protein inference)
- `gene`: character, gene name (symbol) (after protein inference)
- `glycan_composition`: `glyrepr::glycan_composition()`, glycan compositions.
- `glycan_structure`: `glyrepr::glycan_structure()`, glycan structures (if `parse_structure = TRUE`).

Sample information

The sample information file should be a `csv` file with the first column named `sample`, and the rest of the columns being sample information. The `sample` column must match the `RawName` column in the pGlyco3 result file, although the order can be different.

You can put any useful information in the sample information file. Recommended columns are:

- `group`: grouping or conditions, e.g. "control" or "tumor", required for most downstream analyses
- `batch`: batch information, required for batch effect correction

Protein inference

pGlyco3 reports protein groups. That is, shared glycopeptides are reported as a group of proteins separated by ";". This function automatically performs protein inference using the parsimony method to find the leader proteins. This ensures each glycopeptide is uniquely mapped to a single protein, gene, and glycosite.

Aggregation

pGlyco3 performs quantification on the PSM level. This level of information is too detailed for most downstream analyses. This function aggregate PSMs into glycopeptides through summation. For each glycopeptide (unique combination of "peptide", "peptide_site", "protein", "protein_site", "gene", "glycan_composition", "glycan_structure"), we sum up the quantifications of all PSMs that belong to this glycopeptide.

Glycan structures

pGlyco3 reports a "plausible structure" for each glycan. You can set `parse_structure = TRUE` to parse these structures into a "glycan_structure" column as a `glyrepr::glycan_structure()` vector. However, please take caution with these structures, because pGlyco3 does not have strict quality control on glycan structure annotations.

See Also

`glyexp::experiment()`, `glyrepr::glycan_composition()`, `glyrepr::glycan_structure()`

`read_strucgp` *Read the result from StrucGP*

Description

StrucGP is a software for intact glycopeptide identification. As StrucGP doesn't support quantification, this function returns an `glyexp::experiment()` object with a binary (0/1) expression matrix indicating whether each glycopeptide was identified in each sample.

Usage

```
read_strucgp(fp, sample_info = NULL, glycan_type = "N", parse_structure = TRUE)
```

Arguments

`fp`	File path of the StrucGP result file.
`sample_info`	File path of the sample information file (csv), or a sample information data.frame/tibble. If `NULL` (default), a simple sample information tibble will be created.
`glycan_type`	Glycan type. One of "N", "O-GalNAc", "O-GlcNAc", "O-Man", "O-Fuc", or "O-Glc". Default is "N".

`parse_structure`
: Logical. Whether to parse glycan structures. If `TRUE` (default), glycan structures are parsed and included in the `var_info` as `glycan_structure` column. If `FALSE`, structure parsing is skipped and the structure column is removed.

Value

An `glyexp::experiment()` object with a binary (0/1) expression matrix, where 1 indicates the glycopeptide was identified in that sample and 0 indicates it was not.

Variable information

The following columns could be found in the variable information tibble:

- `peptide`: character, peptide sequence
- `protein`: character, protein accession
- `gene`: character, gene name (symbol)
- `protein_site`: integer, site of glycosylation on protein
- `glycan_composition`: `glyrepr::glycan_composition()`, glycan compositions.
- `glycan_structure`: `glyrepr::glycan_structure()`, glycan structures (if `parse_structure = TRUE`).

Sample information

The sample information file should be a `csv` file with the first column named `sample`, and the rest of the columns being sample information. The `sample` column must match the `file_name` column in the StrucGP result file, although the order can be different.

You can put any useful information in the sample information file. Recommended columns are:

- `group`: grouping or conditions, e.g. "control" or "tumor", required for most downstream analyses
- `batch`: batch information, required for batch effect correction

See Also

`glyexp::experiment()`, `glyrepr::glycan_composition()`, `glyrepr::glycan_structure()`

Index