

Package: mighty.metadata (via r-universe)

June 2, 2026

Title Manage 'CDISC' 'ADaM' Dataset Specifications in 'YAML' Format

Version 0.1.0.9003

Description Load, validate, and manipulate Clinical Data Interchange Standards Consortium ('CDISC') Analysis Data Model ('ADaM') dataset metadata stored as 'YAML' files. Metadata files are validated against a JSON schema. Provides functions to inspect and modify columns, parameters, and row-level operations within and across 'ADaM' domains. Designed for use with the 'mighty' framework.

License Apache License (>= 2)

URL <https://novonordisk-opensource.github.io/mighty.metadata/>,
<https://github.com/NovoNordisk-OpenSource/mighty.metadata>

BugReports <https://github.com/NovoNordisk-OpenSource/mighty.metadata/issues>

Depends R (>= 4.1)

Imports cli, dplyr, glue, purrr, rlang, S7, S7schema (>= 0.1.1),
stats, tibble, yaml, zephyr

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

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs libssl-dev libnode-dev

Repository <https://novonordisk-opensource.r-universe.dev>

Date/Publication 2026-06-02 14:31:14 UTC

RemoteUrl <https://github.com/NovoNordisk-OpenSource/mighty.metadata>

RemoteRef HEAD

RemoteSha fd6317f8216a9b00bf2c0da38f4c116c3d42eb9

Contents

columns	2
create_md_col	3
mighty.metadata-options	5
mighty_config	5
mighty_domain	6
mighty_study	7
parameters	9
populate_core	11
populate_sparse	12
resolve_includes	12
rows	13
Index	16

columns	<i>Update columns in your metadata</i>
---------	--

Description

Functions to list, select, remove, add, update, and move columns in your `mighty_domain()` objects.

Usage

```
list_columns(x)
```

```
remove_columns(x, id)
```

```
add_column(x, id, ..., .pos = length(x[["columns"]]) + 1L)
```

```
move_column(x, id, .pos = length(x[["columns"]]))
```

```
select_column(x, id)
```

```
update_column(x, id, ...)
```

Arguments

<code>x</code>	<code>mighty_domain()</code> Object to manipulate.
<code>id</code>	<code>character()</code> Id of the column(s) to remove, add, or move.
<code>...</code>	Additional properties to add/update for the column, e.g. a label = "my label".
<code>.pos</code>	<code>integer(1)</code> Position to put or move the column to. Default places the column as the last.

Value

```
invisible(x)
```

Examples

```
# Load example configuration
x <- mighty_domain(
  file = system.file("examples", "advs.yml", package = "mighty.metadata")
)

# List all columns defined
list_columns(x)

# Remove the STUDYID and USUBJID columns
x |>
  remove_columns(c("STUDYID", "USUBJID")) |>
  list_columns()

# Add a new column
x |>
  add_column(id = "NEW") |>
  list_columns()

# Add new column with label and as the first column
y <- x |>
  add_column(id = "NEW", label = "My label", .pos = 1)

list_columns(y)

y[["columns"]][[1]] |>
  str()

# Move the STUDYID column to the 3rd position
x |>
  move_column(id = "STUDYID", .pos = 3) |>
  list_columns()

# Update an existing column
x |>
  update_column(id = "STUDYID", label = "Updated Label") |>
  select_column(id = "STUDYID") |>
  str()

# Select a specific column
select_column(x, id = "STUDYID") |>
  str()
```

create_md_col

Create Metadata Column Table

Description

Converts a [mighty_study](#) or [mighty_domain](#) object into a flat dataframe of column definitions.

Usage

```
create_md_col(x)
```

Arguments

x A [mighty_study](#) or [mighty_domain](#) object.

Value

A tibble with one row per column containing:

table_id Table identifier

table_label Table label/description

order Column order within table

id Column name

label Column label

origin Origin type (e.g., "Predecessor", "Derived")

key Logical, whether column is a key

is_core Logical, whether column is a core variable

core String, whether a column is Req, Cond or Perm

method Derivation method

codelist Codelist reference

format_type Data type ("C" or "N")

format_length Maximum length

format_display Display format

comment Comment

See Also

[mighty_study](#), [populate_sparse\(\)](#), [populate_core\(\)](#)

Examples

```
study <- mighty_study(  
  path = system.file("examples", package = "mighty.metadata")  
)  
mdcol <- create_md_col(study)
```

 mighty.metadata-options

Options for mighty.metadata

Description

verbosity_level:

Verbosity level for functions in mighty.metadata. See [zephyr::verbosity_level](#) for details.

- Default: NA_character_
- Option: mighty.metadata.verbosity_level
- Environment: R_MIGHTY.METADATA_VERBOSITY_LEVEL

 mighty_config

Mighty Config

Description

mighty_config() provides a robust way of working with the _mighty.yml configuration file in the {mighty} framework.

A new object is initialized by supplying a directory path containing a _mighty.yml file. The file is automatically validated against the mighty.json schema when loaded.

mighty_config() inherits from S7schema::S7schema(). You can validate an object at any time by calling validate() and use write_config() to save it back as a yaml file.

Usage

```
mighty_config(path)
```

Arguments

path character(1) path to a directory containing a _mighty.yml file.

Details

The _mighty.yml file is validated against the mighty.json schema on load. The file must contain an external_data array declaring the primary keys of any datasets external to the ADaM study (e.g. SDTM or reference datasets) that ADaM domain specifications may depend on.

The optional repos field specifies where mighty.component should look for shared components. Each entry is either a local path (e.g. ".") or a GitHub reference in owner/repo/subdir@ref format (e.g. "NovoNordisk-OpenSource/mighty.standards/components@main").

Value

A `mighty_config` S7 object extending [S7schema::S7schema](#).

`external_data` A list of external data source specifications, each with an `id` and `keys` field.

`repos` Optional character vector of component repository locations, or NULL if not specified.

Write Config

Use `write_config()` to serialize a `mighty_config()` object back to a `_mighty.yml` file.

Supply `path` to write to a specific directory; defaults to the directory the object was loaded from.

See Also

[mighty_study](#), [mighty_domain](#), [write_config\(\)](#)

Examples

```
x <- mighty_config(
  path = system.file("examples", package = "mighty.metadata")
)

# Custom print method gives a small overview
print(x)

# Underlying object is a `list`
str(x)

# Write back to a directory
tmp <- tempdir()
write_config(x, path = tmp)
```

mighty_domain

Mighty Domain

Description

`mighty_domain()` provides a robust way of working with ADaM metadata in the `{mighty}` framework.

A new object is initialized by supplying an existing yaml metadata file. This package provides helpers to update column, parameter, and row entries. See the references below for help:

- `help("columns")`
- `help("parameters")`
- `help("rows")`

`mighty_domain()` inherits from `S7schema::S7schema()` and the yaml file is automatically validated when loaded. The helper functions above also always validates the new configuration before returning.

You can at anytime validate an object by calling `validate()` and use `write_config()` to save it as a yaml file again.

Usage

```
mighty_domain(file)
```

Arguments

`file` character(1) path to a yaml file defining an ADaM dataset.

Value

A `mighty_domain` S7 object extending `S7schema::S7schema`. The underlying list contains the parsed and validated YAML metadata including `id`, `label`, `class`, `keys`, `columns`, `parameters`, and `rows`.

Examples

```
x <- mighty_domain(  
  file = system.file("examples", "adv.s.yml", package = "mighty.metadata")  
)  
  
# Custom print method gives a small overview  
print(x)  
  
# Underlying object is a `list`  
str(x)
```

`mighty_study`

Mighty Study

Description

Creates a `mighty_study` object by loading all YAML metadata files from a directory. Each YAML file (except `_mighty.yml` and `_study.yml`) is parsed as a `mighty_domain` object. The optional `_study.yml` file provides study-level properties and the optional `_mighty.yml` file provides mighty framework configuration.

Usage

```
mighty_study(path, populate = FALSE)
```

Arguments

path	character(1) path to a directory containing YAML metadata files.
populate	logical(1) if TRUE, calls <code>populate_core()</code> then <code>populate_sparse()</code> before returning. Default is FALSE.

Details

The function scans the directory for files matching `*.yaml` or `*.yml`:

- Files named `_study.yml` or `_study.yaml` are treated as study properties
- Files named `_mighty.yml` or `_mighty.yaml` are treated as mighty framework config
- All other YAML files must follow ADaM naming conventions (starting with ad) and are loaded as `mighty_domain` objects
- Only one `_mighty.yml` and one `_study.yml` file is allowed per directory

Value

A `mighty_study` S7 object extending `list`:

List elements `mighty_domain` objects, named by their `id` field. Access via e.g. `study$adsl`.

`@study` Study-level properties from `_study.yml`, or empty list if no properties file exists.

`@mighty` A `mighty_config` object loaded from `_mighty.yml`, or NULL if no configuration file exists.

`@path` The source directory path as `character(1)`.

Write Study Metadata

Use `write_config()` to serialize a `mighty_study()` object back to YAML files. Each domain is written as a separate file, plus `_mighty.yml` and `_study.yml` when non-empty.

If `path` is NULL (default), files are written to `x@path`.

See Also

[mighty_domain](#), [write_config\(\)](#), [populate_sparse\(\)](#), [populate_core\(\)](#), [create_md_col\(\)](#)

Examples

```
# Load example study
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)

# List tables with metadata
names(study)

# Access ADVS
study$ADVS
```

```
# Access study-level properties
study@study

# Access mighty framework configuration
study@mighty

# Load and populate in one step
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata"),
  populate = TRUE
)

# Write study back to YAML
tmp <- tempdir()
write_config(study, path = tmp)
```

parameters

Update parameters in your metadata

Description

Functions to list, select, remove, add, update, and move parameters in your `mighty_domain()` objects.

Usage

```
list_parameters(x)

remove_parameters(x, id)

add_parameter(
  x,
  id,
  label,
  columns,
  ...,
  .pos = length(x[["parameters"]]) + 1L
)

move_parameter(x, id, .pos = length(x[["parameters"]]))

update_parameter(x, id, ...)

select_parameter(x, id)
```

Arguments

<code>x</code>	<code>mighty_domain()</code> Object to manipulate.
<code>id</code>	<code>character()</code> Id of the parameter(s) to remove, add, or move.
<code>label</code>	<code>character(1)</code> Parameter label.
<code>columns</code>	<code>list()</code> Columns to set for the parameter.
<code>...</code>	Additional properties to add for the parameter, e.g. a component reference.
<code>.pos</code>	<code>integer(1)</code> Position to put or move the parameter to. Default places the parameter as the last.

Value

`invisible(x)`

Examples

```
# Load example configuration
x <- mighty_domain(
  file = system.file("examples", "adv.s.yml", package = "mighty.metadata")
)

# List all parameters defined
list_parameters(x)

# Remove the BMIGRP parameter
x |>
  remove_parameters("BMIGRP") |>
  list_parameters()

# Add a new parameter
x |>
  add_parameter(
    id = "NEW",
    label = "My new parameter",
    columns = list(list(id = "AVAL"))
  ) |>
  list_parameters()

# Move the BMIGRP parameter to the 1st position
x |>
  move_parameter(id = "BMIGRP", .pos = 1) |>
  list_parameters()

# Update an existing parameter
x |>
  update_parameter(id = "BMI", label = "Updated BMI Label") |>
  select_parameter(id = "BMI") |>
  str()

# Select a specific parameter
select_parameter(x, id = "BMI") |>
```

```
str()
```

populate_core

Populate Core Variables

Description

Adds core variables from supplier datasets as predecessor columns to datasets that use them (marked with `usecore = TRUE`).

Note: Currently only accepts core variables from ADSL.

Usage

```
populate_core(x, ...)
```

Arguments

`x` A [mighty_study](#) or [mighty_domain](#) object.
`...` Additional arguments passed to methods.

Value

A modified [mighty_study](#) or [mighty_domain](#) with core variables added as predecessor columns.

See Also

[mighty_study](#), [populate_sparse\(\)](#), [create_md_col\(\)](#)

Examples

```
study <- mighty_study(  
  path = system.file("examples", package = "mighty.metadata")  
)  
study <- populate_core(study)
```

populate_sparse *Populate Predecessor Metadata*

Description

Populates column metadata from predecessor references. Columns with a method in the format domain.column (e.g., ADSL.USUBJID) inherit metadata from the referenced predecessor.

Usage

```
populate_sparse(x, ...)
```

Arguments

x A [mighty_study](#) or [mighty_domain](#) object.
 ... Additional arguments passed to methods.

Value

A modified [mighty_study](#) or [mighty_domain](#) with predecessor column metadata populated.

See Also

[mighty_study](#), [populate_core\(\)](#), [create_md_col\(\)](#)

Examples

```
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)
study <- populate_sparse(study)
```

resolve_includes *Resolve conditional metadata items*

Description

Evaluates include fields on metadata items (domains, columns, rows, parameters) and removes items where the condition evaluates to FALSE.

Usage

```
resolve_includes(x, info = list())
```

Arguments

x	A <code>mighty_study</code> or <code>mighty_domain</code> object.
info	<code>list()</code> Named list of values used to evaluate include expressions. For <code>mighty_study</code> , merged into <code>x@study</code> .

Details

Include conditions are R expressions with `{glue}` syntax for variable substitution and evaluation. The `info` list provides the values used to do this.

Value

The input object with conditional items resolved.

Examples

```
study <- mighty_study(
  path = system.file("examples", package = "mighty.metadata")
)

# Add a conditional column
study$ADVS <- update_column(
  study$ADVS,
  id = "STUDYID",
  include = "{study_id == 'my_study'}"
)

# Condition TRUE: column kept (study_id is "my_study" in @study)
study |>
  resolve_includes() |>
  getElement("ADVS") |>
  list_columns()

# Condition FALSE: column removed
study |>
  resolve_includes(info = list(study_id = "other")) |>
  getElement("ADVS") |>
  list_columns()
```

rows

Update rows in your metadata

Description

Functions to list, select, remove, add, update, and move row operations in your `mighty_domain()` objects.

Usage

```
list_rows(x)

remove_rows(x, id)

add_row(x, id, ..., .pos = length(x[["rows"]]) + 1L)

move_row(x, id, .pos = length(x[["rows"]]))

update_row(x, id, ...)

select_row(x, id)
```

Arguments

x	mighty_domain() Object to manipulate.
id	character() Id of the row(s) to remove, add, or move.
...	Additional properties to add for the row, e.g. a label = "my row".
.pos	integer(1) Position to put or move the row to. Default places the row as the last.

Value

```
invisible(x)
```

Examples

```
# Load example configuration
x <- mighty_domain(
  file = system.file("examples", "adv.s.yml", package = "mighty.metadata")
)

# List all rows defined
list_rows(x)

# Add a new row
y <- x |>
  add_row(id = "NEW")

list_rows(y)

# Remove the new row again
y |>
  remove_rows("NEW") |>
  list_rows()

# Update an existing row
x |>
  update_row(id = "BASELINE", method = "Updated method") |>
  select_row(id = "BASELINE") |>
```

```
str()

# Select a specific row
select_row(x, id = "BASELINE") |>
  str()
```

Index

`add_column (columns)`, 2
`add_parameter (parameters)`, 9
`add_row (rows)`, 13

`columns`, 2
`create_md_col`, 3
`create_md_col()`, 8, 11, 12

`list_columns (columns)`, 2
`list_parameters (parameters)`, 9
`list_rows (rows)`, 13

`mighty.metadata-options`, 5
`mighty_config`, 5
`mighty_domain`, 3, 4, 6, 6–8, 11–13
`mighty_study`, 3, 4, 6, 7, 11–13
`move_column (columns)`, 2
`move_parameter (parameters)`, 9
`move_row (rows)`, 13

`parameters`, 9
`populate_core`, 11
`populate_core()`, 4, 8, 12
`populate_sparse`, 12
`populate_sparse()`, 4, 8, 11

`remove_columns (columns)`, 2
`remove_parameters (parameters)`, 9
`remove_rows (rows)`, 13
`resolve_includes`, 12
`rows`, 13

`S7schema::S7schema`, 6, 7
`select_column (columns)`, 2
`select_parameter (parameters)`, 9
`select_row (rows)`, 13

`update_column (columns)`, 2
`update_parameter (parameters)`, 9
`update_row (rows)`, 13

`write_config()`, 6, 8
`zephyr::verbosity_level`, 5