Package ‘metacore’

March 31, 2022

Title  A Centralized Metadata Object Focus on Clinical Trial Data Programming Workflows

Version  0.0.4

Description  Create an immutable container holding metadata for the purpose of better enabling programming activities and functionality of other packages within the clinical programming workflow.

License  MIT + file LICENSE

Encoding  UTF-8

RoxygenNote  7.1.2

Suggests  testthat, knitr, rmarkdown, covr

Imports  R6, tidyr, dplyr, stringr, magrittr, XML, purrr, readxl, rlang, tibble, tidyselect

VignetteBuilder  knitr

NeedsCompilation  no

Author  Christina Fillmore [aut, cre] (<https://orcid.org/0000-0003-0595-2302>), Maya Gans [aut] (<https://orcid.org/0000-0002-5452-6089>), Ashley Tarasiewicz [aut], Mike Stackhouse [aut] (<https://orcid.org/0000-0001-6030-723X>), GSK/Atorus JPT [cph, fnd]

Maintainer  Christina Fillmore <christina.e.fillmore@gsk.com>

Depends  R (>= 3.5.0)

Repository  CRAN

Date/Publication  2022-03-31 15:00:02 UTC

\textbf{\texttt{R}} topics documented:

\begin{verbatim}
check_columns  check_structure  check_words  create_tbl  define_to_metacore
\end{verbatim}

\begin{verbatim}
2 3 3 4 4
\end{verbatim}
check_columns

Check all data frames include the correct types of columns

Description

This function checks for vector types and accepted words

Usage

```r
check_columns(
  ds_spec,
  ds_vars,
  var_spec,
  value_spec,
  derivations,
  codelist,
  supp
)
```
check_structure

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ds_spec</td>
<td>dataset specification</td>
</tr>
<tr>
<td>ds_vars</td>
<td>dataset variables</td>
</tr>
<tr>
<td>var_spec</td>
<td>variable specification</td>
</tr>
<tr>
<td>value_spec</td>
<td>value specification</td>
</tr>
<tr>
<td>derivations</td>
<td>derivation information</td>
</tr>
<tr>
<td>codelist</td>
<td>codelist information</td>
</tr>
<tr>
<td>supp</td>
<td>supp information</td>
</tr>
</tbody>
</table>

check_structure  Column Validation Function

Description

Column Validation Function

Usage

check_structure(.data, col, func, any_na_acceptable, nm)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.data</td>
<td>the dataframe to check the column for</td>
</tr>
<tr>
<td>col</td>
<td>the column to test</td>
</tr>
<tr>
<td>func</td>
<td>the function to use to assert column structure</td>
</tr>
<tr>
<td>any_na_acceptable</td>
<td>boolean, testing if the column can have missing</td>
</tr>
<tr>
<td>nm</td>
<td>name of column to check (for warning and error clarification)</td>
</tr>
</tbody>
</table>

check_words  Check Words in Column

Description

Check Words in Column

Usage

check_words(..., col)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>permissible words in the column</td>
</tr>
<tr>
<td>col</td>
<td>the column to check for specific words</td>
</tr>
</tbody>
</table>
### create_tbl

**Create table**

**Description**

This function creates a table from excel sheets. This is mainly used internally for building spec readers, but is exported so others who need to build spec readers can use it.

**Usage**

```r
create_tbl(doc, cols)
```

**Arguments**

- `doc`: list of sheets from a excel doc
- `cols`: vector of regex to get a datasets base on which columns it has. If the vector is named it will also rename the columns

**Value**

- dataset (or list of datasets if not specific enough)

### define_to_metacore

**Define XML to DataDef Object**

**Description**

Given a path, this function converts the define xml to a DataDef Object.

**Usage**

```r
define_to_metacore(path, quiet = FALSE)
```

**Arguments**

- `path`: location of the define xml as a string
- `quiet`: Option to quietly load in, this will suppress warnings, but not errors

**Value**

- DataDef Object
get_control_term  

Get Control Term

Description

Returns the control term (a vector for permitted values and a tibble for code lists) for a given variable. The dataset can be optionally specified if there is different control terminology for different datasets.

Usage

get_control_term(metacode, variable, dataset = NULL)

Arguments

- **metacode**: metacore object
- **variable**: A variable name to get the controlled terms for. This can either be a string or just the name of the variable
- **dataset**: A dataset name. This is not required if there is only one set of control terminology across all datasets

Value

a vector for permitted values and a 2-column tibble for codelists

Examples

```r
meta_ex <- spec_to_metacore(metacore_example("p21_mock.xlsx"))
get_control_term(meta_ex, QVAL, SUPPAE)
get_control_term(meta_ex, "QVAL", "SUPPAE")
```

load_metacore  

load metacore object

Description

load metacore object

Usage

load_metacore(path = NULL)

Arguments

- **path**: location of the metacore object to load into memory
metacore

R6 Class wrapper to create your own metacore object

Description

R6 Class wrapper to create your own metacore object

Usage

```r
metacore(
  ds_spec = tibble(dataset = character(), structure = character(), label = character()),
  ds_vars = tibble(dataset = character(), variable = character(), keep = logical(),
                   key_seq = integer(), order = integer(), core = character(), supp_flag = logical(),
  var_spec = tibble(variable = character(), label = character(), length = integer(),
                     type = character(), common = character(), format = character()),
  value_spec = tibble(dataset = character(), variable = character(), where = character(),
                      type = character(), sig_dig = integer(), code_id = character(), origin = character(),
                     derivation_id = integer(),
  derivations = tibble(derivation_id = integer(), derivation = character()),
  codelist = tibble(code_id = character(), name = character(), type = character(),
                   codes = list()),
  supp = tibble(dataset = character(), variable = character(), idvar = character(),
                qeval = character())
)
```

Arguments

- `ds_spec` contains each dataset in the study, with the labels for each
- `ds_vars` information on what variables are in each dataset + plus dataset specific variable information
- `var_spec` variable information that is shared across all datasets
- `value_spec` parameter specific information, as data is long the specs for wbc might be different the hgb
- `derivations` contains derivation, it allows for different variables to have the same derivation
- `codelist` contains the code/decode information
- `supp` contains the idvar and qeval information for supplemental variables
metacore_example  Get path to metacore example

Description

metacore comes bundled with a number of sample files in its inst/extdata directory. This function make them easy to access. When testing or writing examples in other packages, it is best to use the 'pilot_ADaM.rda' example as it loads fastest.

Usage

metacore_example(file = NULL)

Arguments

file  Name of file. If NULL, the example files will be listed.

Examples

metacore_example()
metacore_example("mock_spec.xlsx")

MetaCore_filter  Select method to subset by a single dataframe

Description

  Select method to subset by a single dataframe

Usage

MetaCore_filter(value)

Arguments

value  the dataframe to subset by
### read_all_sheets

**Description**

Given a path to a file, this function reads in all sheets of an excel file

**Usage**

```r
read_all_sheets(path)
```

**Arguments**

- `path` string of the file path

**Value**

- a list of datasets

### save_metacore

**Description**

save metacore object

**Usage**

```r
save_metacore(metacore_object, path = NULL)
```

**Arguments**

- `metacore_object` the metacore object in memory to save to disc
- `path` file path and file name to save metacore object

**Value**

- an .rda file
select_dataset

Select metacore object to single dataset

Description
Select metacore object to single dataset

Usage
select_dataset(.data, dataset, simplify = FALSE)

Arguments
- .data: the metacore object of dataframes
- dataset: the specific dataset to subset by
- simplify: return a single dataframe

Value
a filtered subset of the metacore object

spec_to_metacore

Specification document to metacore object

Description
This function takes the location of an excel specification document and reads it in as a metacore object. At the moment it only supports specification in the format of pinnacle 21 specifications. But, the @family spec builder can be used as building blocks for bespoke specification documents

Usage
spec_to_metacore(path, quiet = FALSE)

Arguments
- path: string of file location
- quiet: Option to quietly load in, this will suppress warnings, but not errors

Value
given a spec document it returns a metacore object
spec_type

Check the type of spec document

Description

Check the type of spec document

Usage

spec_type(path)

Arguments

path file location as a string

Value

returns string indicating the type of spec document

spec_type_to_codelist

Spec to codelist

Description

Creates the value_spec from a list of datasets (optionally filtered by the sheet input). The named vector *_cols is used to determine which is the correct sheet and renames the columns.

Usage

spec_type_to_codelist(
  doc,
  permitted_val_cols = NULL,
  sheets = NULL,
  simplify = TRUE
)
spec_type_to_derivations

Arguments

- **doc**: Named list of datasets @seealso read_all_sheets() for exact format
- **codelist_cols**: Named vector of column names that make up the codelist. The column names can be regular expressions for more flexibility. But, the names must follow the given pattern
- **permitted_val_cols**: Named vector of column names that make up the permitted value. The column names can be regular expressions for more flexibility. This is optional, can be left as null if there isn’t a permitted value sheet
- **dict_cols**: Named vector of column names that make up the dictionary value. The column names can be regular expressions for more flexibility. This is optional, can be left as null if there isn’t a permitted value sheet
- **sheets**: Optional, regular expressions of the sheets
- **simplify**: Boolean value, if true will convert code/decode pairs that are all equal to a permitted value list

Value

a dataset formatted for the metacore object

See Also

Other spec builder: spec_type_to_derivations(), spec_type_to_ds_spec(), spec_type_to_ds_vars(), spec_type_to_value_spec(), spec_type_to_var_spec()
**Description**

Creates the ds_spec from a list of datasets (optionally filtered by the sheet input). The named vector cols is used to determine which is the correct sheet and renames the columns.

**Usage**

```r
spec_type_to_ds_spec(
  doc,  
  cols = c(dataset = "[N|n]ame|[D|d]ataset|[D|d]omain", structure = "[S|s]tructure", 
            label = "[L|l]abel|[D|d]escription"),  
  sheet = NULL
)
```

**Arguments**

- **doc**
  Named list of datasets @seealso `read_all_sheets()` for exact format
- **cols**
  Named vector of column names. The column names can be regular expressions for more flexibility. But, the names must follow the given pattern
- **sheet**
  Regular expression for the sheet name

**Value**

a dataset formatted for the metacore object

**See Also**

Other spec builder: `spec_type_to_codelist()`, `spec_type_to_ds_spec()`, `spec_type_to_ds_vars()`, `spec_type_to_value_spec()`, `spec_type_to_var_spec()`
Description

Creates the ds_vars from a list of datasets (optionally filtered by the sheet input). The named vector cols is used to determine which is the correct sheet and renames the columns.

Usage

```r
spec_type_to_ds_vars(
  doc,
  key_seq_sep_sheet = TRUE,
  key_seq_cols = c(dataset = "Dataset", key_seq = "Key Variables"),
  sheet = "[V|v]ar|Datasets"
)
```

Arguments

- `doc`: Named list of datasets @seealso `read_all_sheets()` for exact format
- `cols`: Named vector of column names. The column names can be regular expressions for more flexibility. But, the names must follow the given pattern
- `key_seq_sep_sheet`: A boolean to indicate if the key sequence is on a separate sheet. If set to false add the key_seq column name to the cols vector.
- `key_seq_cols`: names vector to get the key_sequence for each dataset
- `sheet`: Regular expression for the sheet names

Value

- a dataset formatted for the metacore object

See Also

Other spec builder: `spec_type_to_codelist()`, `spec_type_to_derivations()`, `spec_type_to_ds_spec()`, `spec_type_to_value_spec()`, `spec_type_to_var_spec()`
Description

Creates the value_spec from a list of datasets (optionally filtered by the sheet input). The named vector cols is used to determine which is the correct sheet and renames the columns.

Usage

```r
spec_type_to_value_spec(
  doc,
  cols = c(dataset = "[D|d]ataset|[D|d]omain", variable = "[N|n]ame|[V|v]ariables?",
            origin = "[O|o]rigin", type = "[T|t]ype", code_id = "[C|c]odelist|Controlled Term",
            sig_dig = "[S|s]ignificant", where = "[W|w]here", derivation_id = "[M|m]ethod"),
  sheet = NULL,
  where_sep_sheet = TRUE,
  where_cols = c(id = "ID", where = c("Variable", "Comparator", "Value")),
  var_sheet = "[V|v]ar"
)
```

Arguments

- **doc**: Named list of datasets @seealso `read_all_sheets()` for exact format
- **cols**: Named vector of column names. The column names can be regular expressions for more flexibility. But, the names must follow the given pattern
- **sheet**: Regular expression for the sheet name
- **where_sep_sheet**: Boolean value to control if the where information in a separate dataset. If the where information is on a separate sheet, set to true and provide the column information with the where_cols inputs.
- **where_cols**: Named list with an id and where field. All columns in the where field will be collapsed together
- **var_sheet**: Name of sheet with the Variable information on it. Metacore expects each variable will have a row in the value_spec. Because many specification only have information in the value tab this is added. If the information already exists in the value tab of your specification set to NULL

Value

a dataset formatted for the metacore object

See Also

Other spec builder: `spec_type_to_codelist()`, `spec_type_to_derivations()`, `spec_type_to_ds_spec()`, `spec_type_to_ds_vars()`, `spec_type_to_var_spec()`
Description

Creates the var_spec from a list of datasets (optionally filtered by the sheet input). The named vector `cols` is used to determine which is the correct sheet and renames the columns. (Note: the keep column will be converted logical)

Usage

```r
spec_type_to_var_spec(
  doc,
  cols = c(variable = "\[N|n\]ame\|\[V|v\]ariables?", length = "\[L|l\]ength", label = "\[L|l\]abel", type = "\[T|t\]ype", dataset = "\[D|d\]ataset\|\[D|d\]omain", format = "\[F|f\]ormat"),
  sheet = NULL
)
```

Arguments

- `doc`  Named list of datasets @seealso `read_all_sheets()` for exact format
- `cols` Named vector of column names. The column names can be regular expressions for more flexibility. But, the names must follow the given pattern
- `sheet` Regular expression for the sheet name

Value

a dataset formatted for the metacore object

See Also

Other spec builder: `spec_type_to_codelist()`, `spec_type_to_derivations()`, `spec_type_to_ds_spec()`, `spec_type_to_ds_vars()`, `spec_type_to_value_spec()`

---

Description

Reads in a define xml and creates a code_list table. The code_list table is a nested tibble where each row is a code list or permitted value list. The code column contains a vector of a tibble depending on if it is a permitted values or code list
xml_to_derivations

Usage

xml_to_derivations(doc)

Arguments

doc xml document

Value

a tibble containing the code list and permitted value information

See Also

Other xml builder: xml_to_codelist(), xml_to_ds_spec(), xml_to_ds_vars(), xml_to_value_spec(), xml_to_var_spec()

xml_to_derivations XML to derivation table

Description

This reads in a xml document and gets all the derivations/comments. These can be cross referenced to variables using the derivation_id's

Usage

xml_to_derivations(doc)

Arguments

doc ?

Value

dataframe with derivation id’s and derivations

See Also

Other xml builder: xml_to_codelist(), xml_to_ds_spec(), xml_to_ds_vars(), xml_to_value_spec(), xml_to_var_spec()
**xml_to_ds_spec**

**XML to Data Set Spec**

**Description**

Creates a dataset specification, which has the domain name and label for each dataset

**Usage**

```r
xml_to_ds_spec(doc)
```

**Arguments**

- `doc`: xml document

**Value**

data frame with the data set specifications

**See Also**

Other xml builder: `xml_to_codelist()`, `xml_to_derivations()`, `xml_to_ds_vars()`, `xml_to_value_spec()`, `xml_to_var_spec()`

---

**xml_to_ds_vars**

**XML to Data Set Var table**

**Description**

Creates the ds_vars table, which acts as a key between the datasets and the var spec

**Usage**

```r
xml_to_ds_vars(doc)
```

**Arguments**

- `doc`: xml document

**Value**

data frame with the dataset and variables

**See Also**

Other xml builder: `xml_to_codelist()`, `xml_to_derivations()`, `xml_to_ds_spec()`, `xml_to_value_spec()`, `xml_to_var_spec()`
xml_to_value_spec  xml to value spec

Description
Takes a define xml and pulls out the value level metadata including codelist_id’s, defines_id’s, and where clause. There is one row per variable expect when there is a where clause, at which point there is one row per value.

Usage
xml_to_value_spec(doc)

Arguments
doc

Value
tibble with the value level information

See Also
Other xml builder: xml_to_codelist(), xml_to_derivations(), xml_to_ds_spec(), xml_to_ds_vars(), xml_to_var_spec()

xml_to_var_spec  XML to variable spec

Description
Takes a define xml and returns a dataset with specifications for each variable. The variable will just be the variable, unless the specification for that variable differ between datasets

Usage
xml_to_var_spec(doc)

Arguments
doc

Value
data frame with variable, length, label columns
See Also

Other xml builder: `xml_to_codelist()`, `xml_to_derivations()`, `xml_to_ds_spec()`, `xml_to_ds_vars()`, `xml_to_value_spec()`
Index

* **Metacore**
  metacore, 6

* **spec builder**
  spec_type_to_codelist, 10
  spec_type_to_derivations, 11
  spec_type_to_ds_spec, 12
  spec_type_to_ds_vars, 13
  spec_type_to_value_spec, 14
  spec_type_to_var_spec, 15

* **xml builder**
  xml_to_codelist, 15
  xml_to_derivations, 16
  xml_to_ds_spec, 17
  xml_to_ds_vars, 17
  xml_to_value_spec, 18
  xml_to_var_spec, 18

check_columns, 2
check_structure, 3
check_words, 3
create_tbl, 4

define_to_metacore, 4

get_control_term, 5

load_metacore, 5

metacore, 6
metacore_example, 7
MetaCore_filter, 7

read_all_sheets, 8
read_all_sheets(), 11–15

save_metacore, 8
select_dataset, 9
spec_to_metacore, 9
spec_type, 10
spec_type_to_codelist, 10, 12–15
spec_type_to_derivations, 11, 12–15
spec_type_to_ds_spec, 11, 12, 13–15
spec_type_to_ds_vars, 11, 12, 13, 14, 15
spec_type_to_value_spec, 11–13, 14, 15
spec_type_to_var_spec, 11–14, 15

xml_to_codelist, 15, 16–19
xml_to_derivations, 16, 16, 17–19
xml_to_ds_spec, 16, 17, 17, 18, 19
xml_to_ds_vars, 16, 17, 17, 18, 19
xml_to_value_spec, 16, 17, 18, 19
xml_to_var_spec, 16–18, 18