Package 'xportr'

September 14, 2023

Title Utilities to Output CDISC SDTM/ADaM XPT Files

Version 0.3.1

Description Tools to build CDISC compliant data sets and check for CDISC compliance.

URL https://github.com/atorus-research/xportr

BugReports https://github.com/atorus-research/xportr/issues

Imports dplyr (>= 1.0.2), purrr (>= 0.3.4), stringr (>= 1.4.0), magrittr, glue (>= 1.4.2), rlang (>= 0.4.10), cli, tidyselect, readr, janitor, tm, haven (>= 2.5.0), lifecycle

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0), withr, knitr, rmarkdown, readxl, DT, labelled, admiral, devtools, spelling, usethis, lintr, metacore

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 3.5)

NeedsCompilation no

Author Eli Miller [aut, cre] (<https://orcid.org/0000-0002-2127-9456>), Vignesh Thanikachalam [aut], Ben Straub [aut], Ross Didenko [aut], Zelos Zhu [aut], Ethan Brockmann [aut], Vedha Viyash [aut], Andre Verissimo [aut], Sophie Shapcott [aut], Celine Piraux [aut], Adrian Chan [aut], Sadchla Mascary [aut], Atorus/GSK JPT [cph]

Maintainer Eli Miller <Eli.Miller@AtorusResearch.com>

Repository CRAN

Date/Publication 2023-09-14 13:30:02 UTC

R topics documented:

`adsl` *Analysis Dataset Subject Level*

Description

An example dataset containing subject level data

Usage

```
adsl
```

Format

`adsl`**:**

A data frame with 254 rows and 48 columns:

STUDYID Study Identifier

USUBJID Unique Subject Identifier

SUBJID Subject Identifier for the Study

SITEID Study Site Identifier

SITEGR1 Pooled Site Group 1

ARM Description of Planned Arm
TRT01P Planned Treatment for Period 01
TRT01PN Planned Treatment for Period 01 (N)
TRT01A Actual Treatment for Period 01
TRT01AN Actual Treatment for Period 01 (N)
TRTSDT Date of First Exposure to Treatment
TRTEDT Date of Last Exposure to Treatment
TRTDUR Duration of Treatment (days)
AVGDD Avg Daily Dose (as planned)
CUMDOSE Cumulative Dose (as planned)
AGE Age
AGEGR1 Pooled Age Group 1
AGEGR1N Pooled Age Group 1 (N)
AGEU Age Units
RACE Race
RACEN Race (N)
SEX Sex
ETHNIC Ethnicity
SAFFL Safety Population Flag
ITTFL Intent-To-Treat Population Flag
EFFFL Efficacy Population Flag
COMP8FL Completers of Week 8 Population Flag
COMP16FL Completers of Week 16 Population Flag
COMP24FL Completers of Week 24 Population Flag
DISCONFL Did the Subject Discontinue the Study
DSRAEFL Discontinued due to AE
DTHFL Subject Died
BMIBL Baseline BMI (kg/m^2)
BMIBLGR1 Pooled Baseline BMI Group 1
HEIGHTBL Baseline Height (cm)
WEIGHTBL Baseline Weight (kg)
EDUCLVL Years of Education
DISONSDT Date of Onset of Disease
DURDIS Duration of Disease (Months)
DURDSGR1 Pooled Disease Duration Group 1
VISIT1DT Date of Visit 1
RFSTDTC Subject Reference Start Date/Time
RFENDTC Subject Reference End Date/Time
VISNUMEN End of Trt Visit (Vis 12 or Early Term.)
RFENDT Date of Discontinuation/Completion
DCDECOD Standardized Disposition Term
DCREASCD Reason for Discontinuation
MMSETOT MMSE Total

label_log *Utility for Variable Labels*

Description

Utility for Variable Labels

Usage

```
label_log(miss_vars, verbose)
```

Arguments

miss_vars	Missing variables in metadata
verbose	Provides additional messaging for user

Value

Output to Console

length_log *Utility for Lengths*

Description

Utility for Lengths

Usage

```
length_log(miss_vars, verbose)
```

Arguments

miss_vars	Variables missing from metadata
verbose	Provides additional messaging for user

Value

Output to Console

type_log *Utility for Types*

Description

Utility for Types

Usage

```
type_log(meta_ordered, type_mismatch_ind, verbose)
```

Arguments

meta_ordered	fill in later
type_mismatch_ind	fill in later
verbose	Provides additional messaging for user

Value

Output to Console

var_names_log *Utility for Renaming Variables*

Description

Utility for Renaming Variables

Usage

```
var_names_log(tidy_names_df, verbose)
```

Arguments

tidy_names_df	dataframe
verbose	Provides additional messaging for user

Value

Output to Console

var_ord_msg *Utility for Ordering*

Description

Utility for Ordering

Usage

```
var_ord_msg(reordered_vars, moved_vars, verbose)
```

Arguments

reordered_vars	Number of variables reordered
moved_vars	Number of variables moved in the dataset
verbose	Provides additional messaging for user

Value

Output to Console

var_spec *Example Dataset Specification*

Description

Example Dataset Specification

Usage

```
var_spec
```

Format

var_spec**:**
A data frame with 216 rows and 19 columns:

Order Order of variable
Dataset Dataset
Variable Variable
Label Variable Label
Data Type Data Type
Length Variable Length
Significant Digits Significant Digits
Format Variable Format

Mandatory Mandatory Variable Flag
Assigned Value Variable Assigned Value
Codelist Variable Codelist
Common Common Variable Flag
Origin Variable Origin
Pages Pages
Method Variable Method
Predecessor Variable Predecessor
Role Variable Role
Comment Comment
Developer Notes Developer Notes

`xportr_df_label` *Assign Dataset Label*

Description

Assigns dataset label from a dataset level metadata to a given data frame. This is stored in the 'label' attribute of the dataframe.

Usage

```
xportr_df_label(.df, metadata = NULL, domain = NULL, metacore = deprecated())
```

Arguments

`.df`	A data frame of CDISC standard.
`metadata`	A data frame containing dataset. See 'Metadata' section for details.
`domain`	Appropriate CDSIC dataset name, e.g. ADAE, DM. Used to subset the metadata object. If none is passed, then name of the dataset passed as .df will be used.
`metacore`	**[Deprecated]** Previously used to pass metadata now renamed with `metadata`

Value

Data frame with label attributes.

Metadata

The argument passed in the 'metadata' argument can either be a metacore object, or a data.frame containing the data listed below. If metacore is used, no changes to options are required.

For data.frame 'metadata' arguments two columns must be present:

1. Domain Name - passed as the 'xportr.df_domain_name' option. Default: "dataset". This is the column subset by the 'domain' argument in the function.
2. Label Name - passed as the 'xportr.df_label' option. Default: "label". Character values to update the 'label' attribute of the dataframe This is passed to `haven::write_xpt` to note the label.

Examples

```
adsl <- data.frame(
  USUBJID = c(1001, 1002, 1003),
  SITEID = c(001, 002, 003),
  AGE = c(63, 35, 27),
  SEX = c("M", "F", "M")
)

metadata <- data.frame(
  dataset = c("adsl", "adae"),
  label = c("Subject-Level Analysis", "Adverse Events Analysis")
)

adsl <- xportr_df_label(adsl, metadata)
```

`xportr_format` *Assign SAS Format*

Description

Assigns a SAS format from a variable level metadata to a given data frame. If no format is found for a given variable, it is set as an empty character vector. This is stored in the format.sas attribute.

Usage

```
xportr_format(.df, metadata = NULL, domain = NULL, metacore = deprecated())
```

Arguments

`.df`	A data frame of CDISC standard.
`metadata`	A data frame containing variable level metadata. See 'Metadata' section for details.
`domain`	Appropriate CDSIC dataset name, e.g. ADAE, DM. Used to subset the metadata object. If none is passed, then name of the dataset passed as .df will be used.
`metacore`	**[Deprecated]** Previously used to pass metadata now renamed with `metadata`

Value

Data frame with `SASformat` attributes for each variable.

Metadata

The argument passed in the 'metadata' argument can either be a metacore object, or a data.frame containing the data listed below. If metacore is used, no changes to options are required.

For data.frame 'metadata' arguments three columns must be present:

1. Domain Name - passed as the 'xportr.domain_name' option. Default: "dataset". This is the column subset by the 'domain' argument in the function.

2. Format Name - passed as the 'xportr.format_name' option. Default: "format". Character values to update the 'format.sas' attribute of the column. This is passed to `haven::write` to note the format.
3. Variable Name - passed as the 'xportr.variable_name' option. Default: "variable". This is used to match columns in '.df' argument and the metadata.

Examples

```
adsl <- data.frame(
  USUBJID = c(1001, 1002, 1003),
  BRTHDT = c(1, 1, 2)
)

metadata <- data.frame(
  dataset = c("adsl", "adsl"),
  variable = c("USUBJID", "BRTHDT"),
  format = c(NA, "DATE9.")
)

adsl <- xportr_format(adsl, metadata)
```

`xportr_label` *Assign Variable Label*

Description

Assigns variable label from a variable level metadata to a given data frame. This function will give detect if a label is greater than 40 characters which isn't allowed in XPT v5. If labels aren't present for the variable it will be assigned an empty character value. Labels are stored in the 'label' attribute of the column.

Usage

```
xportr_label(
  .df,
  metadata = NULL,
  domain = NULL,
  verbose = getOption("xportr.label_verbose", "none"),
  metacore = deprecated()
)
```

Arguments

`.df`	A data frame of CDISC standard.
`metadata`	A data frame containing variable level metadata. See 'Metadata' section for details.
`domain`	Appropriate CDSIC dataset name, e.g. ADAE, DM. Used to subset the metadata object. If none is passed, then name of the dataset passed as .df will be used.

verbose	The action this function takes when an action is taken on the dataset or function validation finds an issue. See 'Messaging' section for details. Options are 'stop', 'warn', 'message', and 'none'
metacore	**[Deprecated]** Previously used to pass metadata now renamed with metadata

Value

Data frame with label attributes for each variable.

Messaging

label_log() is the primary messaging tool for xportr_label(). If there are any columns present in the '.df' that are not noted in the metadata, they cannot be assigned a label and a message will be generated noting the number or variables that have not been assigned a label.

If variables were not found in the metadata and the value passed to the 'verbose' argument is 'stop', 'warn', or 'message', a message will be generated detailing the variables that were missing in metadata.

Metadata

The argument passed in the 'metadata' argument can either be a metacore object, or a data.frame containing the data listed below. If metacore is used, no changes to options are required.

For data.frame 'metadata' arguments three columns must be present:

1. Domain Name - passed as the 'xportr.domain_name' option. Default: "dataset". This is the column subset by the 'domain' argument in the function.
2. Variable Name - passed as the 'xportr.variable_name' option. Default: "variable". This is used to match columns in '.df' argument and the metadata.
3. Variable Label - passed as the 'xportr.label' option. Default: "label". These character values to update the 'label' attribute of the column. This is passed to haven::write to note the label.

Examples

```
adsl <- data.frame(
  USUBJID = c(1001, 1002, 1003),
  SITEID = c(001, 002, 003),
  AGE = c(63, 35, 27),
  SEX = c("M", "F", "M")
)

metadata <- data.frame(
  dataset = "adsl",
  variable = c("USUBJID", "SITEID", "AGE", "SEX"),
  label = c("Unique Subject Identifier", "Study Site Identifier", "Age", "Sex")
)

adsl <- xportr_label(adsl, metadata)
```

`xportr_length` *Assign SAS Length*

Description

Assigns SAS length from a metadata object to a given data frame. If a length isn't present for a variable the length value is set to 200 for character columns, and 8 for non-character columns. This value is stored in the 'width' attribute of the column.

Usage

```
xportr_length(
  .df,
  metadata = NULL,
  domain = NULL,
  verbose = getOption("xportr.length_verbose", "none"),
  metacore = deprecated()
)
```

Arguments

`.df`	A data frame of CDISC standard.
`metadata`	A data frame containing variable level metadata. See 'Metadata' section for details.
`domain`	Appropriate CDSIC dataset name, e.g. ADAE, DM. Used to subset the metadata object. If none is passed, then name of the dataset passed as .df will be used.
`verbose`	The action this function takes when an action is taken on the dataset or function validation finds an issue. See 'Messaging' section for details. Options are 'stop', 'warn', 'message', and 'none'
`metacore`	**[Deprecated]** Previously used to pass metadata now renamed with `metadata`

Value

Data frame with `SASlength` attributes for each variable.

Messaging

`length_log` is the primary messaging tool for `xportr_length`. If there are any columns present in the '.df' that are not noted in the metadata, they cannot be assigned a length and a message will be generated noting the number or variables that have not been assigned a length.

If variables were not found in the metadata and the value passed to the 'verbose' argument is 'stop', 'warn', or 'message', a message will be generated detailing the variables that were missing in the metadata.

Metadata

The argument passed in the 'metadata' argument can either be a {metacore} object, or a data.frame containing the data listed below. If metacore is used, no changes to options are required.

For data.frame 'metadata' arguments three columns must be present:

1. Domain Name - passed as the 'xportr.domain_name' option. Default: "dataset". This is the column subset by the 'domain' argument in the function.
2. Variable Name - passed as the 'xportr.variable_name' option. Default: "variable". This is used to match columns in '.df' argument and the metadata.
3. Variable Label - passed as the 'xportr.length' option. Default: "length". These numeric values to update the 'width' attribute of the column. This is passed to haven::write to note the variable length.

Examples

```
adsl <- data.frame(
  USUBJID = c(1001, 1002, 1003),
  BRTHDT = c(1, 1, 2)
)

metadata <- data.frame(
  dataset = c("adsl", "adsl"),
  variable = c("USUBJID", "BRTHDT"),
  length = c(10, 8)
)

adsl <- xportr_length(adsl, metadata)
```

---

xportr_logger *Utility Logging Function*

Description

Functions to output user messages, usually relating to differences found between dataframe and the metacore/metadata object

Usage

```
xportr_logger(message, type = "none", ...)
```

Arguments

message	Output to be sent out for user
type	Three types: abort, warn, inform
...	additional arguments if needed

Value

Output to Console

`xportr_metadata` *Set variable specifications and domain*

Description

Sets metadata for a dataset in a way that can be accessed by other xportr functions. If used at the start of an xportr pipeline, it removes the need to set metadata and domain at each step individually. For details on the format of the metadata, see the 'Metadata' section for each function in question.

Usage

```
xportr_metadata(.df, metadata, domain = NULL)
```

Arguments

`.df`	A data frame of CDISC standard.
`metadata`	A data frame containing variable level metadata. See 'Metadata' section for details.
`domain`	Appropriate CDSIC dataset name, e.g. ADAE, DM. Used to subset the metadata object. If none is passed, then name of the dataset passed as .df will be used.

Value

`.df` dataset with metadata and domain attributes set

Examples

```
metadata <- data.frame(
  dataset = "test",
  variable = c("Subj", "Param", "Val", "NotUsed"),
  type = c("numeric", "character", "numeric", "character"),
  format = NA,
  order = c(1, 3, 4, 2)
)

adlb <- data.frame(
  Subj = as.character(123, 456, 789),
  Different = c("a", "b", "c"),
  Val = c("1", "2", "3"),
  Param = c("param1", "param2", "param3")
)

xportr_metadata(adlb, metadata, "test")
```

```
if (rlang::is_installed("magrittr")) {
  library(magrittr)

  adlb %>%
    xportr_metadata(metadata, "test") %>%
    xportr_type() %>%
    xportr_order()
}
```

`xportr_order` *Order variables of a dataset according to Spec*

Description

The `dplyr::arrange()` function is used to order the columns of the dataframe. Any variables that are missing an order value are appended to the end of the dataframe after all of the variables that have an order.

Usage

```
xportr_order(
  .df,
  metadata = NULL,
  domain = NULL,
  verbose = getOption("xportr.order_verbose", "none"),
  metacore = deprecated()
)
```

Arguments

`.df`	A data frame of CDISC standard.
`metadata`	A data frame containing variable level metadata. See 'Metadata' section for details.
`domain`	Appropriate CDSIC dataset name, e.g. ADAE, DM. Used to subset the metadata object. If none is passed, then name of the dataset passed as .df will be used.
`verbose`	The action this function takes when an action is taken on the dataset or function validation finds an issue. See 'Messaging' section for details. Options are 'stop', 'warn', 'message', and 'none'
`metacore`	**[Deprecated]** Previously used to pass metadata now renamed with `metadata`

Value

Dataframe that has been re-ordered according to spec

Messaging

`var_ord_msg()` is the primary messaging tool for `xportr_order()`. There are two primary messages that are output from `var_ord_msg()`. The first is the "moved" variables. These are the variables that were not found in the metadata file and moved to the end of the dataset. A message will be generated noting the number, if any, of variables that were moved to the end of the dataset. If any variables were moved, and the 'verbose' argument is 'stop', 'warn', or 'message', a message will be generated detailing the variables that were moved.

The second primary message is the number of variables that were in the dataset, but not in the correct order. A message will be generated noting the number, if any, of variables that have been reordered. If any variables were reordered, and the 'verbose' argument is 'stop', 'warn', or 'message', a message will be generated detailing the variables that were reordered.

Metadata

The argument passed in the 'metadata' argument can either be a metacore object, or a data.frame containing the data listed below. If metacore is used, no changes to options are required.

For data.frame 'metadata' arguments three columns must be present:

1. Domain Name - passed as the 'xportr.domain_name' option. Default: "dataset". This is the column subset by the 'domain' argument in the function.
2. Variable Name - passed as the 'xportr.variable_name' option. Default: "variable". This is used to match columns in '.df' argument and the metadata.
3. Variable Order - passed as the 'xportr.order_name' option. Default: "order". These values used to arrange the order of the variables. If the values of order metadata are not numeric, they will be corsersed to prevent alphabetical sorting of numberic values.

Examples

```
adsl <- data.frame(
  BRTHDT = c(1, 1, 2),
  STUDYID = c("mid987650", "mid987650", "mid987650"),
  TRT01A = c("Active", "Active", "Placebo"),
  USUBJID = c(1001, 1002, 1003)
)

metadata <- data.frame(
  dataset = c("adsl", "adsl", "adsl", "adsl"),
  variable = c("STUDYID", "USUBJID", "TRT01A", "BRTHDT"),
  order = 1:4
)

adsl <- xportr_order(adsl, metadata)
```

xportr_type *Coerce variable type*

Description

XPT v5 datasets only have data types of character and numeric. xportr_type attempts to collapse R classes to those two XPT types. The 'xportr.character_types' option is used to explicitly collapse the class of a column to character using as.character. Similarly, 'xportr.numeric_types' will collapse a column to a numeric type. If no type is passed for a variable and it isn't identifed as a timing variable, it is assumed to be numeric and coerced with as.numeric.

Usage

```
xportr_type(
  .df,
  metadata = NULL,
  domain = NULL,
  verbose = getOption("xportr.type_verbose", "none"),
  metacore = deprecated()
)
```

Arguments

.df	A data frame of CDISC standard.
metadata	A data frame containing variable level metadata. See 'Metadata' section for details.
domain	Appropriate CDSIC dataset name, e.g. ADAE, DM. Used to subset the metadata object. If none is passed, then name of the dataset passed as .df will be used.
verbose	The action this function takes when an action is taken on the dataset or function validation finds an issue. See 'Messaging' section for details. Options are 'stop', 'warn', 'message', and 'none'
metacore	**[Deprecated]** Previously used to pass metadata now renamed with metadata

Details

Certain care should be taken when using timing variables. R serializes dates based on a reference date of 01/01/1970 where XPT uses 01/01/1960. This can result in dates being 10 years off when outputting from R to XPT if you're using a date class. For this reason, xportr will try to determine what should happen with variables that appear to be used to denote time.

For variables that end in DT, DTM, or, TM, if they are not explicitly noted in 'xportr.numeric_types' or 'xportr.character_types', they are coerced to numeric results.

Value

Returns the modified table.

Messaging

`type_log()` is the primary messaging tool for `xportr_type()`. The number of column types that mismatch the reported type in the metadata, if any, is reported by by `xportr_type()`. If there are any type mismatches, and the 'verbose' argument is 'stop', 'warn', or 'message', each mismatch will be detailed with the actual type in the data and the type noted in the metadata.

Metadata

The argument passed in the 'metadata' argument can either be a metacore object, or a data.frame containing the data listed below. If metacore is used, no changes to options are required.

For data.frame 'metadata' arguments four columns must be present:

1. Domain Name - passed as the 'xportr.domain_name' option. Default: "dataset". This is the column subset by the 'domain' argument in the function.
2. Format Name - passed as the 'xportr.format_name' option. Default: "format". Character values to update the 'format.sas' attribute of the column. This is passed to `haven::write` to note the format.
3. Variable Name - passed as the 'xportr.variable_name' option. Default: "variable". This is used to match columns in '.df' argument and the metadata.
4. Variable Type - passed as the 'xportr.type_name'. Default: "type". This is used to note the XPT variable "type" options are numeric or character.
5. (Option only) Character Types - The list of classes that should be explicitly coerced to a XPT Character type. Default: c("character", "char", "text", "date", "posixct", "posixt", "datetime", "time", "partialdate", "partialtime", "partialdatetime", "incompletedatetime", "durationdatetime", "intervaldatetime")
6. (Option only) Numeric Types - The list of classes that should be explicitly coerced to a XPT numeric type. Default: c("integer", "numeric", "num", "float")

Examples

```
metadata <- data.frame(
  dataset = "test",
  variable = c("Subj", "Param", "Val", "NotUsed"),
  type = c("numeric", "character", "numeric", "character"),
  format = NA
)

.df <- data.frame(
  Subj = as.character(123, 456, 789),
  Different = c("a", "b", "c"),
  Val = c("1", "2", "3"),
  Param = c("param1", "param2", "param3")
)

df2 <- xportr_type(.df, metadata, "test")
```

xportr_write *Write xpt v5 transport file*

Description

Writes a local data frame into SAS transport file of version 5. The SAS transport format is an open format, as is required for submission of the data to the FDA.

Usage

```
xportr_write(.df, path, label = NULL, strict_checks = FALSE)
```

Arguments

.df	A data frame to write.
path	Path where transport file will be written. File name sans will be used as xpt name.
label	Dataset label. It must be <=40 characters.
strict_checks	If TRUE, xpt validation will report errors and not write out the dataset. If FALSE, xpt validation will report warnings and continue with writing out the dataset. Defaults to FALSE

Details

- Variable and dataset labels are stored in the "label" attribute.
- SAS length are stored in the "SASlength" attribute.
- SAS format are stored in the "SASformat" attribute.
- SAS type are stored in the "SAStype" attribute.

Value

A data frame. xportr_write() returns the input data invisibly.

Examples

```
adsl <- data.frame(
  Subj = as.character(123, 456, 789),
  Different = c("a", "b", "c"),
  Val = c("1", "2", "3"),
  Param = c("param1", "param2", "param3")
)

xportr_write(adsl,
  path = paste0(tempdir(), "/adsl.xpt"),
  label = "Subject-Level Analysis",
  strict_checks = FALSE
)
```

`xpt_validate` *Validate Dataset Can be Written to xpt*

Description

Function used to validate dataframes before they are sent to `haven::write_xpt` for writing.

Usage

```
xpt_validate(data)
```

Arguments

`data`	Dataset to be exported as xpt file

Value

Returns a character vector of failed conditions

Index