Package 'sdtmval'

October 23, 2023

Title Validate SDTM Domains

Version 0.4.1

Description Provides a set of tools to assist statistical programmers in validating Study Data Tabulation Model (SDTM) domain data sets. Statistical programmers are required to validate that a SDTM data set domain has been programmed correctly, per the SDTM Implementation Guide (SDTMIG) by 'CDISC' (<https://www.cdisc.org/standards/foundational/sdtmig>), study specification, and study protocol using a process called double programming. Double programming involves two different programmers independently converting the raw electronic data cut (EDC) data into a SDTM domain data table and comparing their results to ensure accurate standardization of the data. One of these attempts is termed 'production' and the other 'validation'. Generally, production runs are the official programs for submittals and these are written in 'SAS'. Validation runs can be programmed in another language, in this case 'R'.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

URL https://github.com/skgithub14/sdtmval, https://skgithub14.github.io/sdtmval/

BugReports https://github.com/skgithub14/sdtmval/issues

Imports dplyr, glue, haven, knitr, lubridate, magrittr, purrr, readxl, rlang, stats, stringr, tidyr, tidyselect, utils

Suggests rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 2.10)

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Author Stephen Knapp [aut, cre, cph] (<https://orcid.org/0000-0002-5101-4555>)

Maintainer Stephen Knapp <stephen@knappconsultingllc.com>

Repository CRAN

Date/Publication 2023-10-23 16:10:02 UTC

R topics documented:

`assign_meta_data` *Assign meta data to columns in a SDTM table based on specification file*

Description

Trims the length of each text and date variable to the length specified in the spec and then assigns the attributes `"label"` and `"width"` to each column.

Usage

```
assign_meta_data(
  tbl,
  spec,
  datatype_col = "Data Type",
  var_col = "Variable",
  length_col = "Length",
  label_col = "Label"
)
```

Arguments

tbl	a data frame containing a SDTM table
spec	a data frame with the columns "Variable" which has a value for each column in tbl, "Data Type" which specifies data types by column, "Length" which specifies the character limit for each column, and "Label" which specifies the label for each column
datatype_col	a string, the column in spec that contains the data types (which should include the values "text" and "date"); default is "Data Type"
var_col	a string, the column in spec that contains the domain variable names
length_col	a string, the column in spec that contains the character count limits for each variable
label_col	a string, the column in spec that contains the labels for each variable

Value

a modified copy of tbl with the meta data per specification

See Also

get_data_spec(), get_key_vars(), get_codelist()

Examples

```
work_dir <- system.file("extdata", package = "sdtmval")
spec <- get_data_spec(domain = "XX",
                      dir = work_dir,
                      filename = "spec.xlsx")
after_meta_data <- assign_meta_data(sdtmval::xx_no_meta_data, spec = spec)
labels <- colnames(after_meta_data) |>
  purrr::map(~ attr(after_meta_data[[.]], "label")) |>
  unlist()
lengths <- colnames(after_meta_data) |>
  purrr::map(~ attr(after_meta_data[[.]], "width")) |>
  unlist()
data.frame(
  column = colnames(after_meta_data),
  labels = labels,
  lengths = lengths
```

```
)
```

`assign_SEQ` *Assign SEQ numbers for a SDTM data set*

Description

Assigns the `"[DOMAIN]SEQ"` number by sorting the data set by the specified variables and then grouping by `"USUBJID"`.

Usage

```
assign_SEQ(tbl, key_vars, seq_prefix, USUBJID = "USUBJID")
```

Arguments

`tbl`	a data frame, the SDTM table
`key_vars`	a character vector of the key variables to sort by
`seq_prefix`	a string, the prefix for SEQ as per the spec (usually the two letter domain abbreviation)
`USUBJID`	a string, the column for the subject ID, USUBJID, default is `"USUBJID"`

Value

a sorted copy of the `tbl` data frame with the new SEQ column

Examples

```
df <- data.frame(
  USUBJID = paste("Subject", c(rep(1, 3), rep(2, 3))),
  XXTESTCD = paste("T", rep(c(2, 3, 1), 2))
)
assign_SEQ(df, key_vars = c("USUBJID", "XXTESTCD"), seq_prefix = "XX")
```

calc_DY *Calculate a DY variable (day of study)*

Description

Utilizes the DY method from the SDTM spec: `--DTC-RFSTDTC+1` if `--DTC` is on or after RFSTDTC. `--DTC-RFSTDTC` if `--DTC` precedes `RFSTDTC`. This function can also be used for the ENDY method from the spec which has the same logic.

Usage

```
calc_DY(tbl, DY_col, DTC_col, RFSTDTC = "RFSTDTC")
```

Arguments

`tbl`	a data frame with the date column `RFSTDTC` and the column specified by the `DTC_col` argument
`DY_col`	string, the name of the new DY column to create
`DTC_col`	string, the column in `tbl` which has the dates for which to calculated the DY value; should either already have a date class or be a character vector in the format YYYY-MM-DD
`RFSTDTC`	a string, the column to use for `RFSTDTC`, default is `"RFSTDTC"`; should either already have a date class or be a character vector in the format YYYY-MM-DD

Value

a modified copy of `tbl` with the new DY column

See Also

`create_BLFL()`, `create_EPOCH()`

Examples

```
df <- data.frame(
  DTC = c("2023-08-01", "2023-08-02", "2023-08-03", "2023-08-04"),
  RFSTDTC = rep("2023-08-02", 4)
)
calc_DY(df, DY_col = "XXDY", DTC_col = "DTC")
```

convert_to_script *Convert SDTM QC code from a .Rmd file to .R script*

Description

Wraps `knitr::purl()` to create an .R script from a .Rmd file. It can also auto-archive the .Rmd file to a `[dir]/archive` sub-directory. This is useful for turning first-attempt exploratory data analysis into production scripts once the validation code is complete.

Usage

```
convert_to_script(filename, dir = NULL, archive = FALSE)
```

Arguments

`filename`	string, the file name of both the .Rmd file that will be read and the file name of the .R file to be written (do not include .Rmd or .R extension)
`dir`	string, the directory where the .Rmd file is and the .R file will be written, default is `NULL` which means the current working directory will be used
`archive`	logical, whether to auto-archive the .Rmd file; default is `FALSE`

Details

- The resulting script will take the same name as the .Rmd file but with a different extension (.R)
- If `[dir]/archive` does not already exist, it will be created

Value

nothing

See Also

`write_sessionInfo()`

Examples

```
# get test notebook from the sdtmval/inst/extdata dir and copy it to temp dir
test_file_dir <- system.file("extdata", package = "sdtmval")
filename <- "test_notebook"
temp_path <- tempdir()
file.copy(from = file.path(test_file_dir, paste0(filename, ".Rmd")),
          to = file.path(temp_path, paste0(filename, ".Rmd")))

# create the script and archive the .Rmd file
convert_to_script(dir = temp_path, filename = filename, archive = TRUE)
```

create_BLFL *Create a BLFL column*

Description

Utilizes the BLFL method from the SDTM spec to create a baseline flag: Equal to "Y" for last record with non-missing –ORRES on or before first dose date (RFSTDTC); NA otherwise.

Usage

```
create_BLFL(
  tbl,
  sort_date,
  domain,
  grouping_vars = "USUBJID",
  RFSTDTC = "RFSTDTC",
  compare_date_method = "on or before"
)
```

Arguments

tbl	a data frame with the variables USUBJID, [domain]ORRES, RFSTDTC, and to column specified in the sort_date argument
sort_date	a string, the column name by which to sort records within each USUBJID entry before assigning the BLFL value. This is also the date compared against RFSTDTC to determine the BLFL value. This column should either already have a date class or be a character vector in the format YYYY-MM-DD
domain	a string, the SDTM domain abbreviation
grouping_vars	a character vector of columns to group by when assigning the BLFL, default is "USUBJID". The order of this vector matters.
RFSTDTC	a string, the column to use for RFSTDTC, default is "RFSTDTC"; this columns should either have a date class or a characer class in the YYYY-MM-DD format
compare_date_method	a string, one of c("on or before", "before") indicating where the baseline measurement should be evaluated on or before the study start date or just before; default is "on or before"

Value

a modified copy of tbl with the new column [domain]BLFL

See Also

create_EPOCH(), calc_DY()

Examples

```
df <- dplyr::tibble(
  USUBJID = c(
    rep(1, 3),
    rep(2, 3)
  ),
  XXORRES = c(
    1, 2, 2,
    1, 2, NA
  ),
  XXDTC = as.Date(c(
    "2017-02-05", "2017-02-06", "2017-02-07",
    "2017-02-05", "2017-02-06", "2017-02-07"
  )),
  RFSTDTC = as.Date(c(
    rep("2017-02-05", 3),
    rep("2017-02-07", 3)
  ))
)
create_BLFL(df, sort_date = "XXDTC", domain = "XX")
```

---

create_EPOCH *Create the EPOCH variable*

Description

Utilizes the EPOCH method from the SDTM spec: Missing when `--DTC` is blank; equal to `'SCREENING'` if `--DTC` if before `RFXSTDTC`; equal to `'TREATMENT'` if `--DTC` is on or after `RFXSTDTC` and on or before `RFXENDTC`; equal to `'FOLLOW-UP'` if `--DTC` is after `RFXENDTC`.

Usage

```
create_EPOCH(tbl, date_col, RFXSTDTC = "RFXSTDTC", RFXENDTC = "RFXENDTC")
```

Arguments

`tbl`	a data frame with date class columns `RFXSTDTC` and `RFXENDTC` and the column given in the `date_col` argument
`date_col`	a string, the column name of the event date used to determine the EPOCH; this column can either have a date class or a character class in the YYYY-MM-DD format
`RFXSTDTC`	a string, the date column to use for `RFXSTDTC`, default is `"RFXSTDTC"`; this column can either have a date class or a character class in the YYYY-MM-DD format
`RFXENDTC`	a string, the date column to use for `RFXENDTC`, default is `"RFXENDTC"`; this column can either have a date class or a character class in the YYYY-MM-DD format

Value

a modified copy of `tbl` with the `EPOCH` column

See Also

`create_BLFL()`, `calc_DY()`

Examples

```
df <- data.frame(
  DTC = c("2023-08-01", "2023-08-02", "2023-08-03", "2023-08-04"),
  RFXSTDTC = rep("2023-08-02", 4),
  RFXENDTC = rep("2023-08-03", 4)
)
create_EPOCH(df, date_col = "DTC")
```

---

create_STAT *Assign STAT 'NOT DONE' status*

Description

Creates a –STAT variable and, if all measurements for a visit were not done, also changes all –TESTCD values as "–ALL"

Usage

```
create_STAT(
  df,
  domain,
  nd_ind,
  nd_ind_cd = "Yes",
  USUBJID = "USUBJID",
  VISIT = "VISIT"
)
```

Arguments

`df`	a data frame to modify
`domain`	a string, the domain abbreviation in all caps
`nd_ind`	a string, the variable name in `df` that indicates if a test was not performed, usually a `"Yes"`/`"No"` or `"Y"`/`"N"` column
`nd_ind_cd`	a string, the code from the `nd_ind` column that signifies a test was not done, default is `"Yes"`
`USUBJID`	a string, the variable name in `df` that contains the subject identifier, default is `"USUBJID"`
`VISIT`	a string, the variable name in `df` that indicates a VISIT field, default is `"VISIT"`

Value

a modified copy of df

Examples

```
df <- dplyr::tibble(
  USUBJID = paste("Subject", c(rep("A", 2), rep("B", 4), rep("C", 2))),
  VISIT = paste("Visit", c(1  , 2  , 1  , 1  , 2  , 2  , 2  , 2)),
  XXTESTCD = paste("Test", c(1  , 2  , 1  , 2  , 1  , 2  , 1  , 2)),
  ND = c("N", "N", "Y", "Y", "N", "N", "Y", "Y")
)
create_STAT(df = df, domain = "XX", nd_ind = "ND", nd_ind_cd = "Y")
```

---

dm *Example SDTM Domain 'DM'*

Description

This is an example data set to simulate a SDTM production domain 'DM' which contains study start and end date information by subject. This can be used to test create_BLFL(), create_EPOCH(), and calc_DY().

Usage

```
dm
```

Format

'dm':

A data frame with 2 rows and 4 columns:

USUBJID Subject identifier

RFSTDTC Study start date

RFXSTDTC First exposure date

RFXENDTC Last exposure date

edc_xx *Example EDC data for form/table 'XX'*

Description

This is an example data set to simulate raw EDC data from the fake form/table 'XX'.

Usage

```
edc_xx
```

Format

'edc_xx':
A data frame with 8 rows and 6 columns:

STUDYID Study identifier
USUBJID Subject identifier
VISIT Visit name
XXTESTCD Test name, coded
XXORRES Test result

format_chars_and_dates *Format date and character columns for SDTM tables*

Description

Converts all date columns to character class and replaces all `NA` values in character/date columns with `""`.

Usage

```
format_chars_and_dates(tbl)
```

Arguments

`tbl`	a data frame, the SDTM table

Details

This function should be applied as one of the last steps in the data process but before `assign_meta_data()`.

Value

a modified copy of the `tbl` data frame

See Also

trim_and_make_blanks_NA()

Examples

```
df <- data.frame(
  dates = as.Date(c("2017-02-05", NA)),
  strings = c(NA_character_, "this"),
  nums = c(1, NA)
)
format_chars_and_dates(df)
```

---

get_codelist *Read in the code list from the specification for a specific domain*

Description

Reads-in the "Codelists" sheet from the study's specification MS Excel file and then filters that code list by the variables in the domain

Usage

```
get_codelist(
  domain,
  dir,
  filename,
  var_col = "Variable",
  codelist_sheet = "Codelists",
  varid_col = "ID"
)
```

Arguments

domain	string, SDTM domain or supplemental domain code
dir	string, specification directory
filename	string, file name of the specification
var_col	a string, the column name in the domain spec sheet that contains the variables for that domain, default is "Variable"
codelist_sheet	a string, the sheet name of the spec's code list from the spec's .xlsx file, default is "Codelists"
varid_col	a string, the column name in the codelist_sheet table from the spec's .xlsx file that contains the variable names, default is "ID"

Value

a data frame with the code list

See Also

get_data_spec(), get_key_vars(), assign_meta_data()

Examples

```
work_dir <- system.file("extdata", package = "sdtmval")
codelists <- get_codelist(domain = 'XX',
                          dir = work_dir,
                          filename = "spec.xlsx")
```

---

get_data_spec *Read in the variable specification sheet for a SDTM data set*

Description

Reads the specified domain variable specification sheet from an MS Excel file.

Usage

```
get_data_spec(domain, dir, filename, arrange_by = "Order")
```

Arguments

domain	string, SDTM domain or supplemental domain code
dir	string, specification directory
filename	string, file name of the specification
arrange_by	character vector, the column(s) by which to sort the domain sheet, default is "Order"

Details

The readxl::read_excel() function will causes an access denied warning when reading in a read-only specification file. This does not affect the data import. Variables will be arranged in descending order per the "Order" column in the specification.

Value

a data frame of the variable specification for domain

See Also

get_key_vars(), get_codelist(), assign_meta_data()

Examples

```
work_dir <- system.file("extdata", package = "sdtmval")
spec <- get_data_spec(domain = "XX",
                      dir = work_dir,
                      filename = "spec.xlsx")
```

`get_key_vars` *Read in the key variables for a SDTM domain*

Description

Reads the "Key Variables" column from the SDTM specification MS Excel file's "Datasets" sheet for the specified domain.

Usage

```
get_key_vars(
  domain,
  dir,
  filename,
  datasets_sheet = "Datasets",
  dataset_col = "Dataset",
  keyvar_col = "Key Variables"
)
```

Arguments

domain	string, SDTM domain or supplemental domain code
dir	string, specification directory
filename	string, file name of the specification
datasets_sheet	a string, the sheet name in the specification Excel file that has the key variables, default is "Datasets"
dataset_col	a string, the column name of the domains in the table in datasets_sheet, default is "Dataset"
keyvar_col	a string, the column name of the key variables in the table in datasets_sheet, default is "Key Variables"

Details

The readxl::read_excel() function will causes an access denied warning when reading in a read-only specification file. This does not affect the data import.

Value

a character vector of key variables for the specified domain

See Also

get_data_spec(), get_codelist(), assign_meta_data()

Examples

```
work_dir <- system.file("extdata", package = "sdtmval")
key_vars <- get_key_vars(domain = "XX",
                         dir = work_dir,
                         filename = "spec.xlsx")
```

---

impute_pdates *Impute start or end dates*

Description

Imputes missing date elements for start or end dates. Partial dates should be in the format "UNKN-UN-UN" or some combination of those characters and numbers (ie "2017-UN-UN"). Dates with no information or dates with a missing year will be converted to NA. For start dates, missing days are assumed to be the first of the month while missing months are assumed to be January. For end dates, missing days are assumed to be the last day of the month and missing months are assumed to be December.

Usage

```
impute_pdates(dates, ptype, input_sep = "-")
```

Arguments

dates	a character vector of partial dates (which could also contain full dates) in the format YYYY-MM-DD
ptype	a string of either "start" or "end" indicating whether start or end dates should be imputed, respectively
input_sep	the character that separates date components in dates, default is "-"

Value

a date vector of imputed dates in the format YYYY-MM-DD

See Also

reshape_adates(), reshape_pdates(), trim_dates(), vignette("Dates")

Examples

```
dates <- c(
  "UNKN-UN-UN",
  "2017-UN-UN",
  "2017-02-UN",
  "2017-UN-05",
  "2017-09-03",
  "UNKN-07-14",
  NA
)
impute_pdates(dates, ptype = "start")
impute_pdates(dates, ptype = "end")
```

---

read_edc_tbls *Import EDC data tables*

Description

Reads-in EDC data table .csv files and puts them in a list.

Usage

```
read_edc_tbls(edc_tbls, dir)
```

Arguments

edc_tbls	character vector of EDC table file names (without extension)
dir	string, EDC data directory

Details

The file encoding will be UTF-8.

Value

a named list of data frames where the names are taken from edc_tbls and the data frames are the EDC data tables

See Also

read_sdtm_tbls()

Examples

```
edc_tbls <- c("xx", "vd")
edc_dir <- system.file("extdata", package = "sdtmval")
edc_dat <- read_edc_tbls(edc_tbls, dir = edc_dir)
```

read_sdtm_tbls *Import SDTM data tables*

Description

Reads-in SDTM data tables store as .sas7bdat files and puts them in a list.

Usage

```
read_sdtm_tbls(sdtm_tbls, dir)
```

Arguments

sdtm_tbls	character vector of SDTM table file names (without extension)
dir	string, the directory containing the production SDTM tables

Details

The file encoding will be UTF-8.

Value

a named list of data frames where the names are taken from `sdtm_tbls` and the data frames are the SDTM data

See Also

`read_edc_tbls()`

Examples

```
sdtm_tbls <- "dm"
sdtm_dir <- system.file("extdata", package = "sdtmval")
sdtm_dat <- read_sdtm_tbls(sdtm_tbls, dir = sdtm_dir)
```

reshape_adates *Reshape format of all dates (full and partial)*

Description

Re-arranges full and partial dates in the general form of `"MM/DD/YYYY"` to the ISO 8601 format (`"YYYY-MM-DD"`). This function is appropriate for vectors with mixed full and partial dates because it will not convert the partial dates to `NA` which would occur if you used `as.Date("02/UN/2017", format = "%m/%d/%Y")`.

Usage

```
reshape_adates(dates)
```

Arguments

dates	a character vector of full and/or partial dates

Details

The date component separator in the input vector `dates` can be any character.

Value

a character vector of full and/or partial dates in the format `"YYYY-MM-DD"`

See Also

`reshape_pdates()`, `impute_pdates()`, `trim_dates()`, `vignette("Dates")`

Examples

```
dates <- c("02/05/2017", "UN/UN/2017", "02-05-2017", NA)
reshape_adates(dates)
```

---

reshape_pdates *Reshape format of partial dates*

Description

Re-arranges partial dates from a format of `"UN-UNK-UNKN"` (`"DD-MMM-YYYY"`) to `"UN/UN/UNKN"` (`"MM/DD/YYYY"`).

Usage

```
reshape_pdates(dates, output_sep = "/")
```

Arguments

dates	a character vector of partial dates
output_sep	the date component separator for the output, default is `"/"`

Details

- The separator character between dates components for the input vector `dates` can be any commonly used date separator (`"/"`, `"-"`, `"."`, `" "`).
- In the starting format, the month (`"UNK"`) is a three letter abbreviation but, in the output format, the month is converted to a number
- The output format is a character vector, not a Date vector, to make some common SDTM date workflow operations easier
- The case of the input month abbreviation does not matter; `"Feb"`, `"feb"`, and `"FEB"` will yield the same results

Value

a character vector of partial dates in the format `"UN/UN/UNKN"` (`"MM/DD/YYYY"`)

See Also

`reshape_adates()`, `impute_pdates()`, `trim_dates()`, `vignette("Dates")`

Examples

```
dates <- c(
  "UN-UNK-UNKN",
  "UN/UNK/UNKN",
  "UN-UNK-2017",
  "UN-Feb-2017",
  "05-Feb-2017",
  "05-UNK-2017",
  "05-Feb-UNKN",
  NA
)
reshape_pdates(dates)
```

---

spec_codelists *Example 'Codelists' tab from a SDTM specification .xlsx file*

Description

This table simulates an excerpt from a SDTM specification .xlsx file for the 'Codelists' tab which provides coded and decoded values from `XXTESTCD` and `XXTEST` variables, respectively. This data set can be used to test the `get_codelist()` function.

Usage

```
spec_codelists
```

Format

'spec_codelists':

A data frame with 3 rows and 3 columns:

ID The variable identifier/name

Term The coded term

Decoded Value The corresponding decoded value for the coded term

---

spec_datasets *Example 'Datasets' tab from a SDTM specification .xlsx file*

Description

This table simulates an excerpt from a SDTM specification .xlsx file for the 'Datasets' tab which provides the key variables for the fake domain XX. This data set can be used to test the `get_key_vars()` function.

Usage

```
spec_datasets
```

Format

'spec_datasets':

A data frame with 1 row and 4 columns:

Dataset The domain

Description The domain description

Structure Defines what qualifies as a unique record

Key Variables The domain's key variables

---

spec_XX *Example domain specific tab from a SDTM specification .xlsx file*

Description

This table simulates an excerpt from a SDTM specification .xlsx file for the fake domain tab XX which provides the labels, data types, and lengths by variable. This data set can be used to test the `get_data_spec()` and `assign_meta_data()` functions.

Usage

```
spec_XX
```

Format

'spec_XX':

A data frame with 12 rows and 5 columns:

Order The order of the varibles in the data set

Dataset The domain abbreviation

Varible The domain's variables

Label Variable labels

Data Type Variable data types

Length The maximum allowed length of an entry

`trim_and_make_blanks_NA` *Trim white space and make blanks NA*

Description

Trims the white space on both sides of strings in a character vector and replaces blank values (`""` and `" "`) with `NA` for all columns in a data frame that have a character class.

Usage

```
trim_and_make_blanks_NA(tbl)
```

Arguments

`tbl`	a data frame, the SDTM table

Details

This function should be applied as one of the first steps in the data process to ensure consistent handling of all strings.

Value

a modified copy of the `tbl` data frame

See Also

`format_chars_and_dates()`

Examples

```
df <- data.frame(one = c("   a", "", " "))
trim_and_make_blanks_NA(df)
```

trim_dates *Trim unknown elements in partial dates*

Description

Removes unknown elements from a partial date. For example, "2017-UN-UN" is trimmed to "2017" and "2017-05-UN" is trimmed to "2017-05". Values of "UNKN-UN-UN" are converted to NA. Values where only the year and day are known are converted to just the year, ie "2017-UN-01" converts to "2017". Full dates are not modified.

Usage

```
trim_dates(dates, input_sep = "-")
```

Arguments

dates	a character vector of partial dates in the format "UNKN-UN-UN" ("YYYY-MM-DD"); full dates can also be included
input_sep	the character that separates date components in the input vector dates, default is "-"

Value

a character vector of trimmed partial dates and full dates

See Also

reshape_adates(), reshape_pdates(), impute_pdates(), vignette("Dates")

Examples

```
dates <- c(
  "UNKN-UN-UN",
  "2017-UN-UN",
  "2017-02-UN",
  "2017-UN-05",
  "2017-09-03",
  "UNKN-07-14",
  NA
)
trim_dates(dates)
```

vd *Example EDC data for form/table 'VD'*

Description

This is an example data set to simulate raw EDC data from the 'VD' form/table which contains visit date information by subject.

Usage

```
vd
```

Format

'vd':
A data frame with 6 rows and 3 columns:

USUBJID Subject identifier
VISIT Visit name
VISITDTC Visit date

write_sessionInfo *Write R session information for a script to a .txt file*

Description

Writes a .txt file of the output from `utils::sessionInfo()` with the file name `[filename]_sessionInfo.txt`. By creating a log of the R session conditions a script was run with, results from the script can be reproduced in the future.

Usage

```
write_sessionInfo(filename, dir = NULL)
```

Arguments

`filename`	a string, the script file name (with or without .R extension)
`dir`	a string, the directory to write to, default is `NULL` which means the current working directory will be used

Value

nothing

See Also

`convert_to_script()`

Examples

```
path <- tempdir()
write_sessionInfo(filename = "test.R", dir = path)
```

`write_tbl_to_xpt` *Write a SAS transport file (.xpt)*

Description

Writes a data frame to a SAS transport file (.xpt) named like `"[domain].xpt"`

Usage

```
write_tbl_to_xpt(tbl, filename, dir = NULL, label = NULL)
```

Arguments

`tbl`	a data frame to write
`filename`	a string, the SDTM domain or supplemental domain name which will become the file name and the name attribute of the transport file, the .xpt file extension is optional
`dir`	a string, the directory to write to, default is `NULL`
`label`	a string, the data set name/label for the `haven::write_xpt()` `name` argument. The default is `NULL` in which case the `filename` will be used. `label` must be 8 characters or less.

Details

Files will be written using version 5 .xpt files

Value

nothing

Examples

```
tbl <- dplyr::tibble(one = as.numeric(1:3), two = letters[1:3])
path <- tempdir()
write_tbl_to_xpt(tbl, filename = "test.xpt", dir = path)
```

`xx_no_meta_data` *Example SDTM domain table XX without meta data*

Description

This data set is used to test the `assign_meta_data()` function and contains a fake SDTM domain XX but without label or lengths assigned to the column attributes.

Usage

```
xx_no_meta_data
```

Format

'xx_no_meta_data':

A data frame with 10 rows and 11 columns:

STUDYID Study identifier

USUBJID Subject identifier

XXSEQ Sequence number

XXTESTCD Coded test name

XXTEST Test name

XXORRES Measurement in original units

XXBLFL Baseline flag

VISIT Visit name

EPOCH EPOCH

XXDTC Measurement date

XXDY Measurement day of study

Index