Package ‘EDCimport’

December 19, 2023

Version 0.4.1
Title Import Data from EDC Software
Description A convenient toolbox to import data exported from Electronic Data Capture (EDC) software 'TrialMaster'.
License GPL-3
    https://danchaltiel.github.io/EDCimport/
BugReports https://github.com/DanChaltiel/EDCimport/issues
Depends R (>= 3.1.0)
Imports cli, dplyr, forcats, glue, ggplot2, haven, labelled, purrr, readr, rlang, stringr, tibble, tidy, tidyselect, utils
Suggests callr, gtools, janitor, knitr, plotly, testthat (>= 3.1.8), vdiffir, withr
Encoding UTF-8
RoxygenNote 7.2.3
Config/testthat/edition 3
NeedsCompilation no
Author Dan Chaltiel [aut, cre] (<https://orcid.org/0000-0003-3488-779X>)
Maintainer Dan Chaltiel <dan.chaltiel@gmail.com>
Repository CRAN
Date/Publication 2023-12-19 13:00:03 UTC

R topics documented:

assert_no_duplicate ......................................................... 2
check_subjid ............................................................... 3
data_example ............................................................... 3
edc_options ............................................................... 4
edc Peek options ....................................................... 5
**assert_no_duplicate**

Assert that a dataset has one row per patient

**Description**

Check that there is no duplicate on the column holding patient ID in a pipeable style.
Mostly useful after joining two datasets.

**Usage**

```r
assert_no_duplicate(df, id_col = get_key_cols())
```

**Arguments**

- `df`: the dataset
- `id_col`: (optional) the name of the columns holding patient ID

**Value**

the `df` dataset, unchanged

**Examples**

# without duplicate => no error, continue the pipeline
tibble(subjid=c(1:10)) %>% assert_no_duplicate() %>% nrow()

# with duplicate => throws an error
# tibble(subjid=c(1:10, 1:2)) %>% assert_no_duplicate() %>% nrow()
check_subjid

Check the completion of the subject ID column

Description

Compare a subject ID vector to the study’s reference subject ID (usually something like enrolres$subjid).

Usage

check_subjid(x, ref = getOption("edc_subjid_ref"))

Arguments

x the subject ID column to check
ref the reference for subject ID. Should usually be set through options(edc_subjid_ref=xxx). See example.

Value

nothing, called for warnings

Examples

tm = edc_example()
load_list(tm)
options(edc_subjid_ref=db0$SUBJID)
# usually, you set something like:
# options(edc_subjid_ref=enrolres$subjid)
check_subjid(db1$SUBJID)
check_subjid(db1$SUBJID %>% setdiff(2))
check_subjid(c(db1$SUBJID, 99))

Example databases

Description

List of tables used in EDCimport examples:

- edc_example() can be used as the result of read_trialmaster()
- edc_example.plot() can be used to test edc_swimmerplot()
- edc_example.mixed() can be used to test split_mixed_datasets()
Usage

```r
edc_exampleMixed(N = 100)
edc_examplePlot(N = 50, seed = 42)
edc_example(N = 50, seed = 42)
```

Arguments

- **N**  
  the number of patients
- **seed**  
  the random seed

Value

a list of tables

---

**edc_options**

Set global options for EDCimport

**Description**

Use this function to manage your EDCimport parameters globally while taking advantage of auto-completion.

Use `edc.peek.options()` to see which option is currently set and `edc.reset.options()` to set all options back to default.

**Usage**

```r
edc.options(
  ..., 
  trialmaster_pw, 
  path_7zip, 
  edc_lookup, 
  edc_subjid_ref, 
  edc_plotly, 
  edc_cols_id, 
  edc_cols_crfname, 
  edc_read_verbose, 
  edc_correction_verbose, 
  edc_get_key_cols_verbose, 
  edc_lookup_overwrite_warn, 
  .local = FALSE
)
```
Arguments

... unused
trialmaster_pw the password of the trialmaster zip archive. For instance, you can use `edc_options(trialmaster_pw="my_pwd")` in the console once per session, so that you don’t have to write the password in clear in your R code
path_7zip the path to the 7zip executable. Default to "C:/Program Files/7-Zip/".
edc_lookup (Internal) a reference to the lookup table (usually .lookup). Should usually not be changed manually.
edc_subjid_ref used in `check_subj` the vector of the reference subject IDs. You should usually write `edc_options(edc_subjid_ref=enrolres$subjid)`.
edc_plotly used in `edc_swimmerplot` whether to use plotly to visualize the plot.
edc_cols_id, edc_cols_crfname used in `get_key_cols` the name of the columns holding the subject id (default to c("ptno", "subjid"); and the CRF form name (default to c("crfname"));). It is case-insensitive.
edc_read_verbose, edc_correction_verbose, edc_get_key_cols_verbose the verbosity of the output of functions `read_trialmaster` and `read_tm_all_xpt`, `manual_correction`, and `get_key_cols`. For example, set `edc_options(edc_read_verbose=0)` to silence the first 2.
edc_lookup_overwrite_warn default to TRUE. Whether there should be warning when overwriting .lookup (like when reading 2 databases successively).
.local if TRUE, the effect will only apply to the local frame (internally using `rlang::local_options()`)

Value

Nothing, called for its side effects

---

**edc_peek_options**

See which EDCimport option is currently set.

**Description**

See which EDCimport option is currently set.

**Usage**

`edc_peek_options(keep_null = FALSE)`

**Arguments**

keep_null set to TRUE to get a list

**Value**

A named list of EDCimport options
edc_reset_options  Reset all EDCimport options.

Description
Reset all EDCimport options.

Usage
edc_reset_options(
  except = c("edc_lookup", "trialmaster_pw", "path_7zip"),
  quiet = FALSE
)

Arguments
  except options that are not reset by default
  quiet set to TRUE to remove the message.

Value
Nothing, called for its side effects

edc_swimmerplot  Swimmer plot of all dates columns

Description
Join all tables from .lookup$dataset on id

Usage
edc_swimmerplot(
  .lookup = getOption("edc_lookup"),
  ...,
  id = get_key_cols()$patient_id,
  group = NULL,
  origin = NULL,
  id_lim = NULL,
  exclude = NULL,
  time_unit = c("days", "weeks", "months", "years"),
  aes_color = c("variable", "label"),
  plotly = getOption("edc_plotly", FALSE)
)
### Arguments

- **.lookup**: the lookup table, default to `getOption("edc_lookup")`
- **id**: the patient identifier. Will be coerced as numeric.
- **group**: a grouping variable, given as "dataset$column"
- **origin**: a variable to consider as time 0, given as "dataset$column"
- **id_lim**: a numeric vector of length 2 providing the minimum and maximum id to subset on.
- **exclude**: a character vector of variables to exclude, in the form `dataset$column`. Can be a regex, but $ symbols don’t count. Case-insensitive.
- **time_unit**: if `origin!=NULL`, the unit to measure time. One of c("days", "weeks", "months", "years").
- **aes_color**: either variable "([dataset] - {column})" or label (the column label)
- **plotly**: whether to use {plotly} to get an interactive plot

### Value

Either a plotly or a ggplot

### Examples

```r
tm = read_trialmaster("filename.zip", pw="xx")
tm = edc_example_plot()
load_list(tm)
p = edc_swimmerplot(.lookup, id_lim=c(5,45))
p2 = edc_swimmerplot(.lookup, origin="db0$date_naissance", time_unit="weeks",
          exclude=c("DB1$DATE2", "db3$.x"))
p3 = edc_swimmerplot(.lookup, group="db0$group", aes_color="label")
## Not run:
#save the plotly plot as HTML to share it
htmlwidgets::saveWidget(p, "edc_swimmerplot.html", selfcontained=TRUE)
## End(Not run)
```

---

### Description

This utility extends the lookup table to include:

- **n_id**: the number of patients present in the dataset
- **rows_per_id**: the mean number of row per patient
- **crfname**: the actual name of the dataset
find_keyword

Usage

extend_lookup(
  lookup,
  ...
  ,
  key_columns = get_key-cols(lookup),
  datasets = get_datasets(lookup)
)

Arguments

  lookup [data.frame(1)]
    the lookup table
  ...
    unused
  key_columns [list(n)]
    for experts only
  datasets [data.frame(n)]
    for experts only

Value

  the lookup, extended

Examples

#tm = read_trialmaster("filename.zip", pw="xx")
tm = edc_example_mixed()
load_list(tm)
  lookup
  .lookup = extend_lookup(.lookup)
  .lookup

find_keyword

Find a keyword in the whole database

Description

Find a keyword in all names and labels of a list of datasets.

Usage

find_keyword(keyword, data = getOption("edc_lookup"), ignore_case = TRUE)

Arguments

  keyword the keyword to search for. Can handle regular expressions (see examples).
  data the lookup dataframe where to search the keyword. Can be set using edc_options(edc_lookup=my_data), which is done automatically when calling read_trialmaster().
  ignore_case should case differences be ignored in the match? Default to TRUE.
get_datasets

Value

a tibble

Examples

## Not run:
path = system.file("extdata/Example_Export_SAS_XPORT_2022_08_25_15_16.zip", 
package="EDCimport", mustWork=TRUE)
w = read_trialmaster(path, verbose=FALSE)

find_keyword("patient")

# with regex
find_keyword("patient\$")
find_keyword("\d")
find_keyword("(Trial|Form) Name")
find_keyword("\(\)") # you need to escape special characters

## End(Not run)

get_datasets

Retrieve the datasets as a list of data.frames

Description

Get the datasets from the lookup table as a list of data.frames.

Usage

get_datasets(lookup = getOption("edc_lookup"), envir = parent.frame())

Arguments

lookup the lookup table
envir (internal use)

Value

a list of all datasets
get_key_cols  Important column names

Description
Retrieve names of patient_id (usually "SUBJID" and "PATNO") and crfname (usually "CRF-NAME") from the actual names of the datasets

Usage
get_key_cols(lookup = getOption("edc_lookup"))

Arguments
lookup  the lookup table

Value
a list(2) of characters with names patient_id and crfname

get_lookup  Generate a lookup table

Description
Generate a lookup table

Usage
get_lookup(data_list)

Arguments
data_list  a list containing at least 1 dataframe

Value
a dataframe summarizing column names and labels

Examples
x = edc_example()
x$.lookup=NULL
lk = get_lookup(x)
lk
lk %>% tidyr::unnest(c(names, labels))
load_as_list

Load a .RData file as a list

Description
Instead of loading a .RData file in the global environment, extract every object into a list.

Usage
load_as_list(filename)

Arguments
filename
the filename, with the .RData extension.

Value
a list

Examples
x = list(a=1, b=mtcars)
save_list(x, "test.RData")
y = load_as_list("test.RData")
print(y$a)

load_list

Load a list in an environment

Description
Load a list in an environment

Usage
load_list(x, env = parent.frame(), remove = TRUE)

Arguments
x a list
env the environment onto which the list should be loaded
remove if TRUE, x will be removed from the environment afterward

Value
nothing, called for its side-effect
Examples

```r
x=list(a=1, b=mtcars)
load_list(x, remove=FALSE)
print(a)
print(nrow(b))
```

---

**manual_correction**

**Manual correction**

**Description**

When finding wrong or unexpected values in an exported table, it can be useful to temporarily correct them by hard-coding a value. However, this manual correction should be undone as soon as the central database is updated with the correction.

- `manual_correction()` applies a correction in a specific table column location and throws an error if the correction is already in place. This check applies only once per R session so you can source your script without errors.
- `reset_manual_correction()` resets all checks. For instance, it is called by `read_trialmaster()`.

**Usage**

```r
manual_correction(
  data,
  col,
  rows,
  wrong,
  correct,
  verbose = getOption("edc_correction_verbose", TRUE)
)
reset_manual_correction()
```

**Arguments**

- `data`, `col`, `rows` — the rows of a column of a dataframe where the error lies
- `wrong` — the actual wrong value
- `correct` — the temporary correction value
- `verbose` — whether to print informations (once)

**Value**

Nothing, used for side effects
Examples

library(dplyr)
x = iris %>% mutate(id=row_number(), .before=1) %>% as_tibble()
x$Sepal.Length[c(1,3,5)]

# 1st correction is silent
manual_correction(x, Sepal.Length, rows=c(1,3,5),
                   wrong=c(5.1, 4.7, 5.0), correct=c(5, 4, 3))
x$Sepal.Length[c(1,3,5)]

# further correction is silent
manual_correction(x, Sepal.Length, rows=c(1,3,5),
                   wrong=c(5.1, 4.7, 5.0), correct=c(5, 4, 3))

# if the database is corrected, an error is thrown
## Not run:
reset_manual_correction()
x$Sepal.Length[c(1,3,5)] = c(5, 4, 3) # mimics db correction
manual_correction(x, Sepal.Length, rows=c(1,3,5),
                   wrong=c(5.1, 4.7, 5.0), correct=c(5, 4, 3))

## End(Not run)

---

**read_tm_all_xpt**

Read all .xpt files in a directory

**Description**

Read all .xpt files in a directory (unzipped TrialMaster archive).
If 7zip is installed, you should probably rather use `read_trialmaster()` instead.
If a procformat.sas file exists in the directory, formats will be applied.

**Usage**

```r
read_tm_all_xpt(
directory,
..., 
format_file = "procformat.sas",
clean_names_fun = NULL,
split_mixed = FALSE,
extend_lookup = TRUE,
datetime_extraction = NULL,
verbose = getOption("edc_read_verbose", 1),
key_columns = "deprecated"
"
)```
Arguments

directory [character(1)]
the path to the unzipped archive using SAS_XPORT format. Will read the extraction date from the directory name.

... unused
format_file [character(1)]
the path to the procformat.sas file that should be used to apply formats. Use NULL to not apply formats.
clean_names_fun [function]
a function to clean column names, e.g. tolower, janitor::clean_names()...
split_mixed [logical(1): FALSE]
whether to split mixed datasets. See split_mixed_datasets.
extend_lookup [character(1): FALSE]
whether to enrich the lookup table. See extend_lookup.
datetime_extraction [POSIXt(1)]
the datetime of the data extraction. Default to the most common date of last modification in directory.

verbose [logical(1)]
one of c(0, 1, 2). The higher, the more information will be printed.

key_columns deprecated

Value

a list containing one dataframe for each .xpt file in the folder, the extraction date (datetime_extraction), and a summary of all imported tables (.lookup). If not set yet, option edc_lookup is automatically set to .lookup.

---

read_trialmaster Read the .zip archive of a TrialMaster export

Description

Import the .zip archive of a TrialMaster trial export as a list of dataframes. The archive filename should be leaved untouched as it contains the project name and the date of extraction. Generate a .rds cache file for future reads. If 7zip is not installed or available, use read_tm_all_xpt() instead.
**read_trialmaster**

**Usage**

```r
read_trialmaster(
  archive,
  ...,  
  use_cache = "write",
  clean_names_fun = NULL,
  split_mixed = FALSE,
  extend_lookup = TRUE,
  pw = getOption("trialmaster_pw"),
  verbose = getOption("edc_read_verbose", 1),
  key_columns = "deprecated"
)
```

**Arguments**

- **archive** [character(1)]
  the path to the archive
- **...** unused
- **use_cache** [mixed(1): "write"]
  controls the .rds cache. If TRUE, read the cache if any or extract the archive and create a cache. If FALSE extract the archive without creating a cache file. Can also be "read" or "write".
- **clean_names_fun** [function]
  a function to clean column names, e.g. `tolower, janitor::clean_names()`,...
- **split_mixed** [logical(1): FALSE]
  whether to split mixed datasets. See `split_mixed_datasets`.
- **extend_lookup** [character(1): FALSE]
  whether to enrich the lookup table. See `extend_lookup`.
- **pw** [character(1)]
  The password if the archive is protected. To avoid writing passwords in plain text, it is probably better to use `options(trialmaster_pw="xxx")` instead though.
- **verbose** [logical(1)]
  one of c(0, 1, 2). The higher, the more information will be printed.
- **key_columns** deprecated

**Value**

a list containing one dataframe for each .xpt file in the folder, the extraction date (datetime_extraction), and a summary of all imported tables (.lookup). If not set yet, option edc_lookup is automatically set to .lookup.
save_list  

Save a list as .RData file

Description

Save a list as .RData file

Usage

save_list(x, filename)

Arguments

x a list
filename the filename, with the .RData extension.

Value

nothing, called for its side-effect

Examples

x=list(a=1, b=mtcars)
save_list(x, "test.RData")
load("test.RData")
file.remove("test.RData")
print(a)
print(nrow(b))

split_mixed_datasets  

Split mixed datasets

Description

Split mixed tables, i.e. tables that hold both long data (N values per patient) and short data (one value per patient, duplicated on N lines), into one long table and one short table.

Usage

split_mixed_datasets(
  datasets = get_datasets(),
  id = get_key_cols()$patient_id,
  ..., 
  ignore_cols = getOption("edc_cols_crfname", "CRFNAME"),
  output_code = FALSE,
  verbose = TRUE
)
**unify**

Unify a vector

**Description**

Turn a vector of length N to a vector of length 1 after checking that there is only one unique value. Useful to safely flatten a duplicated table. This preserves the label attribute if set.

**Usage**

```r
unify(x)
```
Arguments

- `x` a vector

Value

- a vector of length 1

Examples

```r
unify(c(1,1,1,1))
#unify(c(1,1,2,1)) #warning

library(dplyr)
x = tibble(id = rep(letters[1:5], 10), value = rep(1:5, 10))
x %>% group_by(id) %>% summarise(value = unify(value)) # safer than `value=value[1]`
x$value[2] = 1
#x %>% group_by(id) %>% summarise(value = unify(value)) # warning about that non-unique value
```
Index

assert_no_duplicate, 2
check_subjid, 3, 5
data_example, 3
edc_example (data_example), 3
edc_example_mixed (data_example), 3
edc_example_plot (data_example), 3
edc_options, 4
edc_peek_options, 5
edc_peek_options(), 4
edc_reset_options, 6
edc_reset_options(), 4
edc_swimmerplot, 5, 6
edc_swimmerplot(), 3
extend_lookup, 7, 14, 15
find_keyword, 8
get_datasets, 9
get_key_cols, 5, 10
get_lookup, 10
janitor::clean_names(), 14, 15
load_as_list, 11
load_list, 11
load_list(), 17
manual_correction, 5, 12
read_tm_all_xpt, 5, 13
read_tm_all_xpt(), 14
read_trialmaster, 5, 14
read_trialmaster(), 3, 8, 12, 13
reset_manual_correction
    (manual_correction), 12
save_list, 16
split_mixed_datasets, 14, 15, 16
tolower, 14, 15
unify, 17