Package 'REDCapCAST'

June 7, 2024

Title REDCap Castellated Data Handling

Version 24.6.1

Description Originally forked from the R part of 'REDCapRITS' by Paul Egeler. See <https://github.com/pegeler/REDCapRITS>. 'REDCap' database casting and handling of castellated data when using repeated instruments and longitudinal projects. Keeps a focused data export approach, by allowing to only export required data from the database. 'REDCap' (Research Electronic Data Capture) is a secure, web-based software platform designed to support data capture for research studies, providing 1) an intuitive interface for validated data capture; 2) audit trails for tracking data manipulation and export procedures; 3) automated export procedures for seamless data downloads to common statistical packages; and 4) procedures for data integration and interoperability with external sources (Harris et al (2009) <doi:10.1016/j.jbi.2008.08.010>; Harris et al (2019) <doi:10.1016/j.jbi.2019.103208>).

Depends R (>= 3.4.0)

Suggests httr, jsonlite, testthat, Hmisc, knitr, rmarkdown, gt, ggplot2, here, styler, devtools, roxygen2, spelling, glue, rhub, shinythemes

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

URL https://github.com/agdamsbo/REDCapCAST, https://agdamsbo.github.io/REDCapCAST/

BugReports https://github.com/agdamsbo/REDCapCAST/issues

Imports dplyr, REDCapR, tidyr, tidyselect, keyring, purrr, readr, stats, shiny, openxlsx2, haven, readODS, zip, assertthat

Collate 'utils.r' 'process_user_input.r' 'REDCap_split.r' 'create_instrument_meta.R' 'doc2dd.R' 'ds2dd.R' 'ds2dd_detailed.R' 'easy_redcap.R' 'html_styling.R'

'mtcars_redcap.R' 'read_redcap_instrument.R'
'read_redcap_tables.R' 'redcap_wider.R' 'redcapcast_data.R'
'redcapcast_meta.R' 'shiny_cast.R'

Language en-US

VignetteBuilder knitr

NeedsCompilation no

Author Andreas Gammelgaard Damsbo [aut, cre]
(<https://orcid.org/0000-0002-7559-1154>),
Paul Egeler [aut] (<https://orcid.org/0000-0001-6948-9498>)

Maintainer Andreas Gammelgaard Damsbo <agdamsbo@clin.au.dk>

Repository CRAN

Date/Publication 2024-06-07 13:00:02 UTC

Contents

case_match_regex_list . . . 3
char2choice . . . 4
char2cond . . . 4
clean_redcap_name . . . 5
create_html_table . . . 6
create_instrument_meta . . . 6
d2w . . . 7
doc2dd . . . 8
ds2dd . . . 9
ds2dd_detailed . . . 11
easy_redcap . . . 12
file_extension . . . 13
focused_metadata . . . 14
format_subheader . . . 14
get_api_key . . . 15
get_id_name . . . 15
guess_time_only_filter . . . 16
hms2character . . . 17
html_tag_wrap . . . 17
is_missing . . . 18
is_repeated_longitudinal . . . 18
mark_complete . . . 19
match_fields_to_form . . . 19
mtcars_redcap . . . 20
process_user_input . . . 20
process_user_input.character . . . 21
process_user_input.data.frame . . . 21
process_user_input.default . . . 22
process_user_input.response . . . 22
read_input . . . 23
read_redcap_instrument . . . 23

read_redcap_tables 24
redcapcast_data 25
redcapcast_meta 26
REDCap_split 27
redcap_wider 29
replace_curly_quote 30
sanitize_split 31
server_factory 32
shiny_cast 32
split_non_repeating_forms 33
strsplitx 34
time_only_correction 34
ui_factory 35

Index **36**

`case_match_regex_list` *List-base regex case_when*

Description

Mimics case_when for list of regex patterns and values. Used for date/time validation generation from name vector. Like case_when, the matches are in order of priority. Primarily used in REDCapCAST to do data type coding from systematic variable naming.

Usage

```
case_match_regex_list(data, match.list, .default = NA)
```

Arguments

`data`	vector
`match.list`	list of case matches
`.default`	Default value for non-matches. Default is NA.

Value

vector

Examples

```
case_match_regex_list(
  c("test_date", "test_time", "test_tida", "test_tid"),
  list(date_dmy = "_dat[eo]$", time_hh_mm_ss = "_ti[md]e?$")
)
```

char2choice *Simple function to generate REDCap choices from character vector*

Description

Simple function to generate REDCap choices from character vector

Usage

```
char2choice(data, char.split = "/", raw = NULL, .default = NA)
```

Arguments

data	vector
char.split	splitting character(s)
raw	specific values. Can be used for options of same length.
.default	default value for missing. Default is NA.

Value

vector

Examples

```
char2choice(c("yes/no","  yep. / nope  ","",NA,"what"),.default=NA)
```

char2cond *Simple function to generate REDCap branching logic from character vector*

Description

Simple function to generate REDCap branching logic from character vector

Usage

```
char2cond(
  data,
  minor.split = ",",
  major.split = ";",
  major.sep = " or ",
  .default = NA
)
```

Arguments

data	vector
minor.split	minor split
major.split	major split
major.sep	argument separation. Default is " or ".
.default	default value for missing. Default is NA.

Value

vector

Examples

```
#data <- dd_inst$betingelse
#c("Extubation_novent, 2; Pacu_delay, 1") |> char2cond()
```

---

clean_redcap_name *clean_redcap_name*

Description

Stepwise removal on non-alphanumeric characters, trailing white space, substitutes spaces for underscores and converts to lower case. Trying to make up for different naming conventions.

Usage

```
clean_redcap_name(x)
```

Arguments

x	vector or data frame for cleaning

Value

vector or data frame, same format as input

`create_html_table` *Create two-column HTML table for data piping in REDCap instruments*

Description

Create two-column HTML table for data piping in REDCap instruments

Usage

```
create_html_table(text, variable)
```

Arguments

`text`	descriptive text
`variable`	variable to pipe

Value

character vector

Examples

```
create_html_table(text = "Patient ID", variable = c("[cpr]"))
create_html_table(text = paste("assessor", 1:2, sep = "_"), variable = c("[cpr]"))
# create_html_table(text = c("CPR nummer","Word"), variable = c("[cpr][1]", "[cpr][2]", "[test]"))
```

`create_instrument_meta` *Create zips file with necessary content based on data set*

Description

Metadata can be added by editing the data dictionary of a project in the initial design phase. If you want to later add new instruments, this can be used to add instrument(s) to a project in production.

Usage

```
create_instrument_meta(data, dir = here::here(""), record.id = TRUE)
```

Arguments

`data`	metadata for the relevant instrument. Could be from ‘ds2dd_detailed()‘
`dir`	destination dir for the instrument zip. Default is the current WD.
`record.id`	flag to omit the first row of the data dictionary assuming this is the record_id field which should not be included in the instrument. Default is TRUE.

Value

list

Examples

```
data <- iris |>
  ds2dd_detailed(add.auto.id = TRUE,
  form.name=sample(c("b","c"),size = 6,replace = TRUE,prob=rep(.5,2))) |>
  purrr::pluck("meta")
# data |> create_instrument_meta()

data <- iris |>
  ds2dd_detailed(add.auto.id = FALSE) |>
  purrr::pluck("data")
names(data) <- glue::glue("{sample(x = c('a','b'),size = length(names(data)),
replace=TRUE,prob = rep(x=.5,2))}__{names(data)}")
data <- data |> ds2dd_detailed(form.sep="__")
# data |>
#   purrr::pluck("meta") |>
#   create_instrument_meta(record.id = FALSE)
```

---

d2w *Convert single digits to words*

Description

Convert single digits to words

Usage

```
d2w(x, lang = "en", neutrum = FALSE, everything = FALSE)
```

Arguments

x	data. Handle vectors, data.frames and lists
lang	language. Danish (da) and English (en), Default is "en"
neutrum	for numbers depending on counted word
everything	flag to also split numbers >9 to single digits

Value

returns characters in same format as input

Examples

```
d2w(c(2:8, 21))
d2w(data.frame(2:7, 3:8, 1), lang = "da", neutrum = TRUE)

## If everything=T, also larger numbers are reduced.
## Elements in the list are same length as input
d2w(list(2:8, c(2, 6, 4, 23), 2), everything = TRUE)
```

---

doc2dd *Doc table to data dictionary - EARLY, DOCS MISSING*

Description

Works well with 'project.aid::docx2list()'. Allows defining a database in a text document (see provided template) for an easier to use data base creation. This approach allows easier collaboration when defining the database. The generic case is a data frame with variable names as values in a column. This is a format like the REDCap data dictionary, but gives a few options for formatting.

Usage

```
doc2dd(
  data,
  instrument.name,
  col.variables = 1,
 list.datetime.format = list(date_dmy = "_dat[eo]$", time_hh_mm_ss = "_ti[md]e?$"),
  col.description = NULL,
  col.condition = NULL,
  col.subheader = NULL,
  subheader.tag = "h2",
  condition.minor.sep = ",",
  condition.major.sep = ";",
  col.calculation = NULL,
  col.choices = NULL,
  choices.char.sep = "/",
  missing.default = NA
)
```

Arguments

`data`	tibble or data.frame with all variable names in one column
`instrument.name`	character vector length one. Instrument name.
`col.variables`	variable names column (default = 1), allows dplyr subsetting
`list.datetime.format`	formatting for date/time detection. See 'case_match_regex_list()'

col.description
: descriptions column, allows dplyr subsetting. If empty, variable names will be used.

col.condition
: conditions for branching column, allows dplyr subsetting. See ‘char2cond()‘.

col.subheader
: sub-header column, allows dplyr subsetting. See ‘format_subheader()‘.

subheader.tag
: formatting tag. Default is "h2"

condition.minor.sep
: condition split minor. See ‘char2cond()‘. Default is ",".

condition.major.sep
: condition split major. See ‘char2cond()‘. Default is ";".

col.calculation
: calculations column. Has to be written exact. Character vector.

col.choices
: choices column. See ‘char2choice()‘.

choices.char.sep
: choices split. See ‘char2choice()‘. Default is "/".

missing.default
: value for missing fields. Default is NA.

Value

tibble or data.frame (same as data)

Examples

```
# data <- dd_inst
# data |> doc2dd(instrument.name = "evt",
# col.description = 3,
# col.condition = 4,
# col.subheader = 2,
# col.calculation = 5,
# col.choices = 6)
```

---

ds2dd *(DEPRECATED) Data set to data dictionary function*

Description

Creates a very basic data dictionary skeleton. Please see ‘ds2dd_detailed()‘ for a more advanced function.

Usage

```
ds2dd(
  ds,
  record.id = "record_id",
  form.name = "basis",
  field.type = "text",
  field.label = NULL,
  include.column.names = FALSE,
  metadata = metadata_names
)
```

Arguments

ds	data set
record.id	name or column number of id variable, moved to first row of data dictionary, character of integer. Default is "record_id".
form.name	vector of form names, character string, length 1 or length equal to number of variables. Default is "basis".
field.type	vector of field types, character string, length 1 or length equal to number of variables. Default is "text.
field.label	vector of form names, character string, length 1 or length equal to number of variables. Default is NULL and is then identical to field names.
include.column.names	Flag to give detailed output including new column names for original data set for upload.
metadata	Metadata column names. Default is the included REDCapCAST::metadata_names.

Details

Migrated from stRoke ds2dd(). Fits better with the functionality of 'REDCapCAST'.

Value

data.frame or list of data.frame and vector

Examples

```
redcapcast_data$record_id <- seq_len(nrow(redcapcast_data))
ds2dd(redcapcast_data, include.column.names=TRUE)
```

ds2dd_detailed *Extract data from stata file for data dictionary*

Description

Extract data from stata file for data dictionary

Usage

```
ds2dd_detailed(
  data,
  add.auto.id = FALSE,
  date.format = "dmy",
  form.name = NULL,
  form.sep = NULL,
  form.prefix = TRUE,
  field.type = NULL,
  field.label = NULL,
  field.label.attr = "label",
  field.validation = NULL,
  metadata = names(REDCapCAST::redcapcast_meta),
  validate.time = FALSE,
  time.var.sel.pos = "[Tt]i[d(me)]",
  time.var.sel.neg = "[Dd]at[eo]"
)
```

Arguments

data	data frame
add.auto.id	flag to add id column
date.format	date format, character string. ymd/dmy/mdy. dafault is dmy.
form.name	manually specify form name(s). Vector of length 1 or ncol(data). Default is NULL and "data" is used.
form.sep	If supplied dataset has form names as suffix or prefix to the column/variable names, the seperator can be specified. If supplied, the form.sep is ignored. Default is NULL.
form.prefix	Flag to set if form is prefix (TRUE) or suffix (FALSE) to the column names. Assumes all columns have pre- or suffix if specified.
field.type	manually specify field type(s). Vector of length 1 or ncol(data). Default is NULL and "text" is used for everything but factors, which wil get "radio".
field.label	manually specify field label(s). Vector of length 1 or ncol(data). Default is NULL and colnames(data) is used or attribute ‘field.label.attr‘ for haven_labelled data set (imported .dta file with ‘haven::read_dta()‘).

`field.label.attr`
: attribute name for named labels for haven_labelled data set (imported .dta file with ‘haven::read_dta()‘. Default is "label"

`field.validation`
: manually specify field validation(s). Vector of length 1 or ncol(data). Default is NULL and ‘levels()‘ are used for factors or attribute ‘factor.labels.attr‘ for haven_labelled data set (imported .dta file with ‘haven::read_dta()‘).

`metadata`
: redcap metadata headings. Default is REDCapCAST:::metadata_names.

`validate.time`
: Flag to validate guessed time columns

`time.var.sel.pos`
: Positive selection regex string passed to ‘gues_time_only_filter()‘ as sel.pos.

`time.var.sel.neg`
: Negative selection regex string passed to ‘gues_time_only_filter()‘ as sel.neg.

Details

This function is a natural development of the ds2dd() function. It assumes that the first column is the ID-column. No checks. Please, do always inspect the data dictionary before upload.

Ensure, that the data set is formatted with as much information as possible.

‘field.type‘ can be supplied

Value

list of length 2

Examples

```
data <- REDCapCAST::redcapcast_data
data |> ds2dd_detailed(validate.time = TRUE)
data |> ds2dd_detailed()
iris |> ds2dd_detailed(add.auto.id = TRUE)
mtcars |> ds2dd_detailed(add.auto.id = TRUE)
data <- iris |>
  ds2dd_detailed(add.auto.id = TRUE) |>
  purrr::pluck("data")
names(data) <- glue::glue("{sample(x = c('a','b'),size = length(names(data)),
replace=TRUE,prob = rep(x=.5,2))}__{names(data)}")
data |> ds2dd_detailed(form.sep="__")
```

---

easy_redcap *Secure API key storage and data acquisition in one*

Description

Secure API key storage and data acquisition in one

Usage

```
easy_redcap(project.name, widen.data = TRUE, uri, ...)
```

Arguments

project.name	The name of the current project (for key storage with 'keyring::key_set()', using the default keyring)
widen.data	argument to widen the exported data
uri	REDCap database API uri
...	arguments passed on to 'REDCapCAST::read_redcap_tables()'

Value

data.frame or list depending on widen.data

---

file_extension *Helper to import files correctly*

Description

Helper to import files correctly

Usage

```
file_extension(filenames)
```

Arguments

filenames	file names

Value

character vector

Examples

```
file_extension(list.files(here::here(""))[[2]])[[1]]
file_extension(c("file.cd..ks","file"))
```

focused_metadata *focused_metadata*

Description

Extracts limited metadata for variables in a dataset

Usage

```
focused_metadata(metadata, vars_in_data)
```

Arguments

metadata	A dataframe containing metadata
vars_in_data	Vector of variable names in the dataset

Value

A dataframe containing metadata for the variables in the dataset

format_subheader *Sub-header formatting wrapper*

Description

Sub-header formatting wrapper

Usage

```
format_subheader(data, tag = "h2")
```

Arguments

data	character vector
tag	character vector length 1

Value

character vector

Examples

```
"Instrument header" |> format_subheader()
```

get_api_key *Retrieve project API key if stored, if not, set and retrieve*

Description

Retrieve project API key if stored, if not, set and retrieve

Usage

```
get_api_key(key.name)
```

Arguments

key.name	character vector of key name

Value

character vector

get_id_name *Get the id name*

Description

Get the id name

Usage

```
get_id_name(data)
```

Arguments

data	data frame or list

Value

character vector

guess_time_only_filter *Try at determining which are true time only variables*

Description

This is just a try at guessing data type based on data class and column names hoping for a tiny bit of naming consistency. R does not include a time-only data format natively, so the "hms" class from ‘readr‘ is used. This has to be converted to character class before REDCap upload.

Usage

```
guess_time_only_filter(
  data,
  validate = FALSE,
  sel.pos = "[Tt]i[d(me)]",
  sel.neg = "[Dd]at[eo]"
)
```

Arguments

data	data set
validate	flag to output validation data. Will output list.
sel.pos	Positive selection regex string
sel.neg	Negative selection regex string

Value

character vector or list depending on ‘validate‘ flag.

Examples

```
data <- redcapcast_data
data |> guess_time_only_filter()
data |>
  guess_time_only_filter(validate = TRUE) |>
  lapply(head)
```

hms2character *Change "hms" to "character" for REDCap upload.*

Description

Change "hms" to "character" for REDCap upload.

Usage

```
hms2character(data)
```

Arguments

data	data set

Value

data.frame or tibble

Examples

```
data <- redcapcast_data
## data |> time_only_correction() |> hms2character()
```

html_tag_wrap *Simple html tag wrapping for REDCap text formatting*

Description

Simple html tag wrapping for REDCap text formatting

Usage

```
html_tag_wrap(data, tag = "h2", extra = NULL)
```

Arguments

data	character vector
tag	character vector length 1
extra	character vector

Value

character vector

Examples

```
html_tag_wrap("Titel", tag = "div", extra = 'class="rich-text-field-label"')
html_tag_wrap("Titel", tag = "h2")
```

`is_missing` *Multi missing check*

Description

Multi missing check

Usage

```
is_missing(data, nas = c("", "NA"))
```

Arguments

`data`	character vector
`nas`	character vector of strings considered as NA

Value

logical vector

`is_repeated_longitudinal` *Test if repeatable or longitudinal*

Description

Test if repeatable or longitudinal

Usage

```
is_repeated_longitudinal(
  data,
 generics = c("redcap_event_name", "redcap_repeat_instrument", "redcap_repeat_instance")
)
```

Arguments

`data`	data set
`generics`	default is "redcap_event_name", "redcap_repeat_instrument" and "redcap_repeat_instance"

Value

logical

Examples

```
is_repeated_longitudinal(c("record_id", "age", "record_id", "gender"))
is_repeated_longitudinal(redcapcast_data)
is_repeated_longitudinal(list(redcapcast_data))
```

---

mark_complete *Completion marking based on completed upload*

Description

Completion marking based on completed upload

Usage

```
mark_complete(upload, ls)
```

Arguments

upload	output list from 'REDCapR::redcap_write()'
ls	output list from 'ds2dd_detailed()'

Value

list with 'REDCapR::redcap_write()' results

---

match_fields_to_form *Match fields to forms*

Description

Match fields to forms

Usage

```
match_fields_to_form(metadata, vars_in_data)
```

Arguments

metadata	A data frame containing field names and form names
vars_in_data	A character vector of variable names

Value

A data frame containing field names and form names

mtcars_redcap *mtcars dataset slightly modified to use for Shiny app upload demonstration*

Description

mtcars dataset slightly modified to use for Shiny app upload demonstration

Usage

```
data(mtcars_redcap)
```

Format

A data frame with 13 variables:

record_id ID, numeric

mpg ID, numeric

cyl ID, numeric

disp ID, numeric

hp ID, numeric

drat ID, numeric

wt ID, numeric

qsec ID, numeric

vs ID, numeric

am ID, numeric

gear ID, numeric

carb ID, numeric

name original rownames, charater

process_user_input *User input processing*

Description

User input processing

Usage

```
process_user_input(x)
```

Arguments

x	input

Value

processed input

---

process_user_input.character
User input processing character

Description

User input processing character

Usage

```
## S3 method for class 'character'
process_user_input(x, ...)
```

Arguments

x	input
...	ignored

Value

processed input

---

process_user_input.data.frame
User input processing data.frame

Description

User input processing data.frame

Usage

```
## S3 method for class 'data.frame'
process_user_input(x, ...)
```

Arguments

x	input
...	ignored

Value

processed input

`process_user_input.default`
User input processing default

Description

User input processing default

Usage

```
## Default S3 method:
process_user_input(x, ...)
```

Arguments

`x`	input
`...`	ignored

Value

processed input

`process_user_input.response`
User input processing response

Description

User input processing response

Usage

```
## S3 method for class 'response'
process_user_input(x, ...)
```

Arguments

`x`	input
`...`	ignored

Value

processed input

`read_input` *Flexible file import based on extension*

Description

Flexible file import based on extension

Usage

```
read_input(file, consider.na = c("NA", "\"\"", ""))
```

Arguments

`file`	file name
`consider.na`	character vector of strings to consider as NAs

Value

tibble

Examples

```
read_input("https://raw.githubusercontent.com/agdamsbo/cognitive.index.lookup/main/data/sample.csv")
```

`read_redcap_instrument` *Convenience function to download complete instrument, using token storage in keyring.*

Description

Convenience function to download complete instrument, using token storage in keyring.

Usage

```
read_redcap_instrument(
  key,
  uri,
  instrument,
  raw_or_label = "raw",
  id_name = "record_id",
  records = NULL
)
```

Arguments

key	key name in standard keyring for token retrieval.
uri	REDCap database API uri
instrument	instrument name
raw_or_label	raw or label passed to 'REDCapR::redcap_read()'
id_name	id variable name. Default is "record_id".
records	specify the records to download. Index numbers. Numeric vector.

Value

data.frame

---

read_redcap_tables *Download REDCap data*

Description

Implementation of REDCap_split with a focused data acquisition approach using REDCapR::redcap_read and only downloading specified fields, forms and/or events using the built-in focused_metadata including some clean-up. Works with classical and longitudinal projects with or without repeating instruments.

Usage

```
read_redcap_tables(
  uri,
  token,
  records = NULL,
  fields = NULL,
  events = NULL,
  forms = NULL,
  raw_or_label = "label",
  split_forms = "all"
)
```

Arguments

uri	REDCap database API uri
token	API token
records	records to download
fields	fields to download
events	events to download
forms	forms to download
raw_or_label	raw or label tags
split_forms	Whether to split "repeating" or "all" forms, default is all.

Value

list of instruments

Examples

```
# Examples will be provided later
```

---

redcapcast_data *Data set for demonstration*

Description

This is a small dataset from a REDCap database for demonstrational purposes. Contains only synthetic data.

Usage

```
data(redcapcast_data)
```

Format

A data frame with 22 variables:

record_id ID, numeric

redcap_event_name Event name, character

redcap_repeat_instrument Repeat instrument, character

redcap_repeat_instance Repeat instance, numeric

cpr CPR number, character

inclusion Inclusion date, Date

inclusion_time Inclusion time, hms

dob Date of birth, Date

age Age decimal, numeric

age_integer Age integer, numeric

sex Legal sex, character

cohabitation Cohabitation status, character

hypertension Hypertension, character

diabetes diabetes, character

region region, character

baseline_data_start_complete Completed, character

mrs_assessed mRS Assessed, character

mrs_date Assesment date, Date

mrs_score Categorical score, numeric

mrs_complete Complete, numeric

event_datetime Event datetime, POSIXct

event_age Age at time of event, numeric

event_type Event type, character

new_event_complete Completed, character

`redcapcast_meta` *REDCap metadata from data base*

Description

This metadata dataset from a REDCap database is for demonstrational purposes.

Usage

```
data(redcapcast_meta)
```

Format

A data frame with 22 variables:

field_name field_name, character

form_name form_name, character

section_header section_header, character

field_type field_type, character

field_label field_label, character

select_choices_or_calculations select_choices_or_calculations, character

field_note field_note, character

text_validation_type_or_show_slider_number text_validation_type_or_show_slider_number, character

text_validation_min text_validation_min, character

text_validation_max text_validation_max, character

identifier identifier, character

branching_logic branching_logic, character

required_field required_field, character

custom_alignment custom_alignment, character

question_number question_number, character

matrix_group_name matrix_group_name, character

matrix_ranking matrix_ranking, character

field_annotation field_annotation, character

REDCap_split *Split REDCap repeating instruments table into multiple tables*

Description

This will take output from a REDCap export and split it into a base table and child tables for each repeating instrument. Metadata is used to determine which fields should be included in each resultant table.

Usage

```
REDCap_split(
  records,
  metadata,
  primary_table_name = "",
  forms = c("repeating", "all")
)
```

Arguments

records	Exported project records. May be a data.frame, response, or character vector containing JSON from an API call.
metadata	Project metadata (the data dictionary). May be a data.frame, response, or character vector containing JSON from an API call.
primary_table_name	Name given to the list element for the primary output table (as described in *README.md*). Ignored if forms = 'all'.
forms	Indicate whether to create separate tables for repeating instruments only or for all forms.

Value

A list of "data.frame"s. The number of tables will differ depending on the forms option selected.

- 'repeating': one base table and one or more tables for each repeating instrument.
- 'all': a data.frame for each instrument, regardless of whether it is a repeating instrument or not.

Author(s)

Paul W. Egeler, M.S., GStat

Examples

```
## Not run:
# Using an API call ----------------------------------------------------

library(RCurl)

# Get the records
records <- postForm(
  uri = api_url, # Supply your site-specific URI
  token = api_token, # Supply your own API token
  content = "record",
  format = "json",
  returnFormat = "json"
)

# Get the metadata
metadata <- postForm(
  uri = api_url, # Supply your site-specific URI
  token = api_token, # Supply your own API token
  content = "metadata",
  format = "json"
)

# Convert exported JSON strings into a list of data.frames
REDCapRITS::REDCap_split(records, metadata)

# Using a raw data export ----------------------------------------------

# Get the records
records <- read.csv("/path/to/data/ExampleProject_DATA_2018-06-03_1700.csv")

# Get the metadata
metadata <- read.csv(
  "/path/to/data/ExampleProject_DataDictionary_2018-06-03.csv"
)

# Split the tables
REDCapRITS::REDCap_split(records, metadata)

# In conjunction with the R export script ------------------------------

# You must set the working directory first since the REDCap data export
# script contains relative file references.
old <- getwd()
setwd("/path/to/data/")

# Run the data export script supplied by REDCap.
# This will create a data.frame of your records called 'data'
source("ExampleProject_R_2018-06-03_1700.r")

# Get the metadatan
metadata <- read.csv("ExampleProject_DataDictionary_2018-06-03.csv")
```

```
# Split the tables
REDCapRITS::REDCap_split(data, metadata)
setwd(old)

## End(Not run)
```

---

redcap_wider *Redcap Wider*

Description

Converts a list of REDCap data frames from long to wide format. Handles longitudinal projects, but not yet repeated instruments.

Usage

```
redcap_wider(
  data,
  event.glue = "{.value}_{redcap_event_name}",
  inst.glue = "{.value}_{redcap_repeat_instance}"
)
```

Arguments

data	A list of data frames.
event.glue	A dplyr::glue string for repeated events naming
inst.glue	A dplyr::glue string for repeated instruments naming

Value

The list of data frames in wide format.

Examples

```
# Longitudinal
list1 <- list(
  data.frame(
    record_id = c(1, 2, 1, 2),
    redcap_event_name = c("baseline", "baseline", "followup", "followup"),
    age = c(25, 26, 27, 28)
  ),
  data.frame(
    record_id = c(1, 2),
    redcap_event_name = c("baseline", "baseline"),
    gender = c("male", "female")
  )
)
redcap_wider(list1)
```

```
# Simpel with two instruments
list2 <- list(
  data.frame(
    record_id = c(1, 2),
    age = c(25, 26)
  ),
  data.frame(
    record_id = c(1, 2),
    gender = c("male", "female")
  )
)
redcap_wider(list2)
# Simple with single instrument
list3 <- list(data.frame(
  record_id = c(1, 2),
  age = c(25, 26)
))
redcap_wider(list3)
# Longitudinal with repeatable instruments
list4 <- list(
  data.frame(
    record_id = c(1, 2, 1, 2),
    redcap_event_name = c("baseline", "baseline", "followup", "followup"),
    age = c(25, 26, 27, 28)
  ),
  data.frame(
    record_id = c(1, 1, 1, 1, 2, 2, 2, 2),
    redcap_event_name = c(
      "baseline", "baseline", "followup", "followup",
      "baseline", "baseline", "followup", "followup"
    ),
    redcap_repeat_instrument = "walk",
    redcap_repeat_instance = c(1, 2, 1, 2, 1, 2, 1, 2),
    dist = c(40, 32, 25, 33, 28, 24, 23, 36)
  ),
  data.frame(
    record_id = c(1, 2),
    redcap_event_name = c("baseline", "baseline"),
    gender = c("male", "female")
  )
)
redcap_wider(list4)
```

`replace_curly_quote` *Replace curly apostrophes and quotes from word*

Description

Copied from textclean, which has not been updated since 2018 and is not on CRAN. Github:https://github.com/trinker/textclea

Usage

```
replace_curly_quote(x)
```

Arguments

x	character vector

Value

character vector

`sanitize_split` *Sanitize list of data frames*

Description

Removing empty rows

Usage

```
sanitize_split(
  l,
  generic.names = c("redcap_event_name", "redcap_repeat_instrument",
    "redcap_repeat_instance")
)
```

Arguments

l	A list of data frames.
generic.names	A vector of generic names to be excluded.

Value

A list of data frames with generic names excluded.

server_factory *Shiny server factory*

Description

Shiny server factory

Usage

```
server_factory()
```

Value

shiny server

shiny_cast *Launch the included Shiny-app for database casting and upload*

Description

Launch the included Shiny-app for database casting and upload

Usage

```
shiny_cast()
```

Value

shiny app

Examples

```
# shiny_cast()
```

`split_non_repeating_forms` *Split a data frame into separate tables for each form*

Description

Split a data frame into separate tables for each form

Usage

```
split_non_repeating_forms(table, universal_fields, fields)
```

Arguments

`table`	A data frame
`universal_fields`	A character vector of fields that should be included in every table
`fields`	A two-column matrix containing the names of fields that should be included in each form

Value

A list of data frames, one for each non-repeating form

Examples

```
# Create a table
table <- data.frame(
  id = c(1, 2, 3, 4, 5),
  form_a_name = c("John", "Alice", "Bob", "Eve", "Mallory"),
  form_a_age = c(25, 30, 25, 15, 20),
  form_b_name = c("John", "Alice", "Bob", "Eve", "Mallory"),
  form_b_gender = c("M", "F", "M", "F", "F")
)

# Create the universal fields
universal_fields <- c("id")

# Create the fields
fields <- matrix(
  c(
    "form_a_name", "form_a",
    "form_a_age", "form_a",
    "form_b_name", "form_b",
    "form_b_gender", "form_b"
  ),
  ncol = 2, byrow = TRUE
)
```

```
# Split the table
split_non_repeating_forms(table, universal_fields, fields)
```

`strsplitx` *Extended string splitting*

Description

Can be used as a substitute of the base function. Main claim to fame is easing the split around the defined delimiter, see example.

Usage

```
strsplitx(x, split, type = "classic", perl = FALSE, ...)
```

Arguments

`x`	data
`split`	delimiter
`type`	Split type. Can be c("classic", "before", "after", "around")
`perl`	perl param from strsplit()
`...`	additional parameters are passed to base strsplit handling splits

Value

list

Examples

```
test <- c("12 months follow-up", "3 steps", "mRS 6 weeks",
"Counting to 231 now")
strsplitx(test, "[0-9]", type = "around")
```

`time_only_correction` *Correction based on time_only_filter function*

Description

Correction based on time_only_filter function

Usage

```
time_only_correction(data, ...)
```

Arguments

data	data set
...	arguments passed on to ‘guess_time_only_filter()‘

Value

tibble

Examples

```
data <- redcapcast_data
## data |> time_only_correction()
```

---

ui_factory *UI factory for shiny app*

Description

UI factory for shiny app

Usage

```
ui_factory()
```

Value

shiny ui

Index

∗ **datasets**
    mtcars_redcap, 20
    redcapcast_data, 25
    redcapcast_meta, 26

case_match_regex_list, 3
char2choice, 4
char2cond, 4
clean_redcap_name, 5
create_html_table, 6
create_instrument_meta, 6

d2w, 7
doc2dd, 8
ds2dd, 9
ds2dd_detailed, 11

easy_redcap, 12

file_extension, 13
focused_metadata, 14
format_subheader, 14

get_api_key, 15
get_id_name, 15
guess_time_only_filter, 16

hms2character, 17
html_tag_wrap, 17

is_missing, 18
is_repeated_longitudinal, 18

mark_complete, 19
match_fields_to_form, 19
mtcars_redcap, 20

process_user_input, 20
process_user_input.character, 21
process_user_input.data.frame, 21
process_user_input.default, 22
process_user_input.response, 22

read_input, 23
read_redcap_instrument, 23
read_redcap_tables, 24
REDCap_split, 27
redcap_wider, 29
redcapcast_data, 25
redcapcast_meta, 26
replace_curly_quote, 30

sanitize_split, 31
server_factory, 32
shiny_cast, 32
split_non_repeating_forms, 33
strsplitx, 34

time_only_correction, 34

ui_factory, 35