

Package ‘REDCapCAST’

February 28, 2024

Title REDCap Castellated Data Handling

Version 24.2.1

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](https://doi.org/10.1016/j.jbi.2008.08.010)>; Harris et al (2019) <[doi:10.1016/j.jbi.2019.103208](https://doi.org/10.1016/j.jbi.2019.103208)>).

Depends R (>= 3.4.0)

Suggests httr, jsonlite, testthat, Hmisc, knitr, rmarkdown, gt, usethis, ggplot2, here, styler, devtools, roxygen2, spelling

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, rsconnect, haven

Collate 'utils.r' 'process_user_input.r' 'REDCap_split.r' 'ds2dd.R' 'ds2dd_detailed.R' 'easy_redcap.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 2024-02-28 07:30:02 UTC

R topics documented:

clean_redcap_name	3
d2w	3
deploy_shiny	4
ds2dd	4
ds2dd_detailed	6
easy_redcap	7
file_extension	8
focused_metadata	8
get_api_key	9
get_id_name	9
guess_time_only_filter	10
hms2character	11
is_repeated_longitudinal	11
mark_complete	12
match_fields_to_form	12
mtcars_redcap	13
read_input	13
read_redcap_instrument	14
read_redcap_tables	15
redcapcast_data	16
redcapcast_meta	17
REDCap_split	18
redcap_wider	20
sanitize_split	21
server_factory	22
shiny_cast	22
split_non_repeating_forms	23
strsplitx	24
time_only_correction	25
ui_factory	25

Index 26

clean_redcap_name	<i>clean_redcap_name</i>
-------------------	--------------------------

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

d2w	<i>Convert single digits to words</i>
-----	---------------------------------------

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)
```

deploy_shiny	<i>Deploy the Shiny app with rsconnect</i>
--------------	--

Description

Deploy the Shiny app with rsconnect

Usage

```
deploy_shiny(path = here::here("app/"), name.app = "shiny_cast")
```

Arguments

path	app folder path
name.app	name of deployed app

Value

deploy

Examples

```
# deploy_shiny
```

ds2dd	<i>(DEPRECATED) Data set to data dictionary function</i>
-------	--

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,
  field.type = NULL,
  field.label = NULL,
  field.label.attr = "label",
  field.validation = NULL,
  metadata = metadata_names,
  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. default is dmy.
form.name	manually specify form name(s). Vector of length 1 or ncol(data). Default is NULL and "data" is used.
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 will 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_data
data |> ds2dd_detailed(validate.time = TRUE)
data |> ds2dd_detailed()
iris |> ds2dd_detailed(add.auto.id = TRUE)
mtcars |> ds2dd_detailed(add.auto.id = TRUE)
  
```

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

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

Value

data.frame or list depending on `widen.data`

file_extension	<i>Helper to import files correctly</i>
----------------	---

Description

Helper to import files correctly

Usage

```
file_extension(filenamees)
```

Arguments

filenamees file names

Value

character vector

Examples

```
file_extension(list.files(here::here(""))[[2]])[[1]]
```

focused_metadata	<i>focused_metadata</i>
------------------	-------------------------

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

get_api_key	<i>Retrieve project API key if stored, if not, set and retrieve</i>
-------------	---

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	<i>Get the id name</i>
-------------	------------------------

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

<code>data</code>	data set
<code>validate</code>	flag to output validation data. Will output list.
<code>sel.pos</code>	Positive selection regex string
<code>sel.neg</code>	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	<i>Change "hms" to "character" for REDCap upload.</i>
---------------	---

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()
```

is_repeated_longitudinal	<i>Test if repeatable or longitudinal</i>
--------------------------	---

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	<i>Completion marking based on completed upload</i>
---------------	---

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	<i>Match fields to forms</i>
----------------------	------------------------------

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	<i>mtcars dataset slightly modified to use for Shiny app upload demonstration</i>
---------------	---

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, character

read_input	<i>Flexible file import based on extension</i>
------------	--

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	<i>REDCap metadata from data base</i>
-----------------	---------------------------------------

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

<code>records</code>	Exported project records. May be a <code>data.frame</code> , response, or character vector containing JSON from an API call.
<code>metadata</code>	Project metadata (the data dictionary). May be a <code>data.frame</code> , response, or character vector containing JSON from an API call.
<code>primary_table_name</code>	Name given to the list element for the primary output table (as described in <i>README.md</i>). Ignored if <code>forms = 'all'</code> .
<code>forms</code>	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 metadata
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

<code>data</code>	A list of data frames.
<code>event.glue</code>	A dplyr::glue string for repeated events naming
<code>inst.glue</code>	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)
```

```

# Simple 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)

```

sanitize_split

Sanitize list of data frames

Description

Removing empty rows

Usage

```
sanitize_split(
  1,
  generic.names = c("redcap_event_name", "redcap_repeat_instrument",
                    "redcap_repeat_instance")
)
```

Arguments

1 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	<i>Shiny server factory</i>
----------------	-----------------------------

Description

Shiny server factory

Usage

```
server_factory()
```

Value

shiny server

shiny_cast	<i>Launch the included Shiny-app for database casting and upload</i>
------------	--

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](#), 13
- [redcapcast_data](#), 16
- [redcapcast_meta](#), 17

[clean_redcap_name](#), 3

[d2w](#), 3
[deploy_shiny](#), 4
[ds2dd](#), 4
[ds2dd_detailed](#), 6

[easy_redcap](#), 7

[file_extension](#), 8
[focused_metadata](#), 8

[get_api_key](#), 9
[get_id_name](#), 9
[guess_time_only_filter](#), 10

[hms2character](#), 11

[is_repeated_longitudinal](#), 11

[mark_complete](#), 12
[match_fields_to_form](#), 12
[mtcars_redcap](#), 13

[read_input](#), 13
[read_redcap_instrument](#), 14
[read_redcap_tables](#), 15
[REDCap_split](#), 18
[redcap_wider](#), 20
[redcapcast_data](#), 16
[redcapcast_meta](#), 17

[sanitize_split](#), 21
[server_factory](#), 22
[shiny_cast](#), 22
[split_non_repeating_forms](#), 23

[strsplitx](#), 24

[time_only_correction](#), 25

[ui_factory](#), 25