

Package ‘REDCapTidieR’

July 21, 2025

Type Package

Title Extract 'REDCap' Databases into Tidy 'Tibble's

Version 1.2.3

Description Convert 'REDCap' exports into tidy tables for easy handling of 'REDCap' repeat instruments and event arms.

License MIT + file LICENSE

URL <https://chop-cgtinformatics.github.io/REDCapTidieR/>,
<https://github.com/CHOP-CGTInformatics/REDCapTidieR>

BugReports <https://github.com/CHOP-CGTInformatics/REDCapTidieR/issues>

Depends R (>= 3.5.0)

Imports checkmate, cli, dplyr, glue, lobstr, lubridate, purrr, REDCapR
(>= 1.2.0), rlang, stringi, stringr, tibble, tidyr, tidysselect,
formattable, pillar, vctrs, readr, stats, forcats

Suggests covr, knitr, labelled, lintr, openxlsx2 (>= 0.8),
prettyunits, rmarkdown, skimr, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Richard Hanna [aut, cre] (ORCID:
<<https://orcid.org/0009-0005-6496-8154>>),
Stephan Kadauke [aut] (ORCID: <<https://orcid.org/0000-0003-2996-8034>>),
Ezra Porter [aut] (ORCID: <<https://orcid.org/0000-0002-4690-8343>>)

Maintainer Richard Hanna <hannar1@chop.edu>

Repository CRAN

Date/Publication 2025-06-06 16:20:05 UTC

Contents

add_skimr_metadata	2
bind_tibbles	3
combine_checkboxes	4
extract_tibble	6
extract_tibbles	7
format-helpers	8
make_labelled	9
read_redcap	10
superheroes_supertbl	12
tbl_sum.redcap_supertbl	13
vec_ptype_abbr.redcap_supertbl	13
write_redcap_xlsx	14
Index	16

add_skimr_metadata	Add <i>skimr</i> metrics to a supertibble’s metadata
--------------------	--

Description

Add default *skim* metrics to the redcap_data list elements of a supertibble output from read_redcap.

Usage

```
add_skimr_metadata(supertbl)
```

Arguments

supertbl a supertibble generated using read_redcap()

Details

For more information on the default metrics provided, check the [get_default_skimmer_names](#) documentation.

Value

A supertibble with *skimr* metadata metrics

Examples

```
superheroes_supertbl

add_skimr_metadata(superheroes_supertbl)

## Not run:
redcap_uri <- Sys.getenv("REDCAP_URI")
```

```
token <- Sys.getenv("REDCAP_TOKEN")

supertbl <- read_redcap(redcap_uri, token)
add_skimr_metadata(supertbl)

## End(Not run)
```

bind_tibbles	<i>Extract data tibbles from a REDCapTidieR supertibble and bind them to an environment</i>
--------------	---

Description

Take a supertibble generated with `read_redcap()` and bind its data tibbles (i.e. the tibbles in the `redcap_data` column) to an environment. The default is the global environment.

Usage

```
bind_tibbles(supertbl, environment = global_env(), tbls = NULL)
```

Arguments

<code>supertbl</code>	A supertibble generated by <code>read_redcap()</code> . Required.
<code>environment</code>	The environment to bind the tibbles to. Default is <code>rlang::global_env()</code> .
<code>tbls</code>	A vector of the <code>redcap_form_names</code> of the data tibbles to bind to the environment. Default is <code>NULL</code> which binds all data tibbles.

Value

This function returns nothing as it's used solely for its side effect of modifying an environment.

Examples

```
## Not run:
# Create an empty environment
my_env <- new.env()

ls(my_env)

superheroes_supertbl

bind_tibbles(superheroes_supertbl, my_env)

ls(my_env)

## End(Not run)
```

combine_checkboxes	<i>Combine Checkbox Fields into a Single Column</i>
--------------------	---

Description

`combine_checkboxes()` consolidates multiple checkbox fields in a REDCap data tibble into a single column. This transformation simplifies analysis by merging several binary columns into one labeled factor column, making the data more interpretable and easier to analyze.

Usage

```
combine_checkboxes(
  supertbl,
  tbl,
  cols,
  names_prefix = "",
  names_sep = "_",
  names_glue = NULL,
  names_repair = "check_unique",
  multi_value_label = "Multiple",
  values_fill = NA,
  raw_or_label = "label",
  keep = TRUE
)
```

Arguments

<code>supertbl</code>	A supertibble generated by <code>read_redcap()</code> . Required.
<code>tbl</code>	The <code>redcap_form_name</code> of the data tibble to extract. Required.
<code>cols</code>	Checkbox columns to combine to single column. Required.
<code>names_prefix</code>	String added to the start of every variable name.
<code>names_sep</code>	String to separate new column names from <code>names_prefix</code> .
<code>names_glue</code>	Instead of <code>names_sep</code> and <code>names_prefix</code> , you can supply a glue specification and the unique .value to create custom column names.
<code>names_repair</code>	What happens if the output has invalid column names? The default, "check_unique" is to error if the columns are duplicated. Use "minimal" to allow duplicates in the output, or "unique" to de-duplicated by adding numeric suffixes. See <code>vctrs::vec_as_names()</code> for more options.
<code>multi_value_label</code>	A string specifying the value to be used when multiple checkbox fields are selected. Default "Multiple".
<code>values_fill</code>	Value to use when no checkboxes are selected. Default NA.
<code>raw_or_label</code>	Either 'raw' or 'label' to specify whether to use raw coded values or labels for the options. Default 'label'.
<code>keep</code>	Logical indicating whether to keep the original checkbox fields in the output. Default TRUE.

Details

`combine_checkboxes()` operates on the data and metadata tibbles produced by the `read_redcap()` function. Since it relies on the checkbox field naming conventions used by REDCap, changes to the checkbox variable names or their associated metadata `field_names` could lead to errors.

REDCap checkbox fields are typically expanded into separate variables for each checkbox option, with names formatted as `checkbox_var___1`, `checkbox_var___2`, etc. `combine_checkboxes()` detects these variables and combines them into a single column. If the expected variables are not found, an error is returned.

Value

A modified supertibble.

Examples

```
library(dplyr)
# Set up sample data tibble
data_tbl <- tibble::tribble(
  ~"study_id", ~"multi___1", ~"multi___2", ~"multi___3",
  1, TRUE, FALSE, FALSE,
  2, TRUE, TRUE, FALSE,
  3, FALSE, FALSE, FALSE
)

# Set up sample metadata tibble
metadata_tbl <- tibble::tribble(
  ~"field_name", ~"field_type", ~"select_choices_or_calculations",
  "study_id", "text", NA,
  "multi___1", "checkbox", "1, Red | 2, Yellow | 3, Blue",
  "multi___2", "checkbox", "1, Red | 2, Yellow | 3, Blue",
  "multi___3", "checkbox", "1, Red | 2, Yellow | 3, Blue"
)

# Create sample supertibble
supertbl <- tibble::tribble(
  ~"redcap_form_name", ~"redcap_data", ~"redcap_metadata",
  "tbl", data_tbl, metadata_tbl
)

class(supertbl) <- c("redcap_supertbl", class(supertbl))

# Combine checkboxes under column "multi"
combine_checkboxes(
  supertbl = supertbl,
  tbl = "tbl",
  cols = starts_with("multi")
) %>%
  dplyr::pull(redcap_data) %>%
  dplyr::first()

## Not run:
```

```

redcap_uri <- Sys.getenv("REDCAP_URI")
token <- Sys.getenv("REDCAP_TOKEN")

supertbl <- read_redcap(redcap_uri, token)
combine_checkboxes(
  supertbl = supertbl,
  tbl = "tbl",
  cols = starts_with("col"),
  multi_value_label = "Multiple",
  values_fill = NA
)

## End(Not run)

```

extract_tibble

Extract a single data tibble from a REDCapTidieR supertibble

Description

Take a supertibble generated with `read_redcap()` and return one of its data tibbles.

Usage

```
extract_tibble(supertbl, tbl)
```

Arguments

<code>supertbl</code>	A supertibble generated by <code>read_redcap()</code> . Required.
<code>tbl</code>	The <code>redcap_form_name</code> of the data tibble to extract. Required.

Details

This function makes it easy to extract a single instrument's data from a REDCapTidieR supertibble.

Value

A tibble.

Examples

```

superheroes_supertbl

extract_tibble(superheroes_supertbl, "heroes_information")

```

extract_tibbles	<i>Extract data tibbles from a REDCapTidieR supertibble into a list</i>
-----------------	---

Description

Take a supertibble generated with `read_redcap()` and return a named list of data tibbles.

Usage

```
extract_tibbles(supertbl, tbls = everything())
```

Arguments

<code>supertbl</code>	A supertibble generated by <code>read_redcap()</code> . Required.
<code>tbls</code>	A vector of <code>form_names</code> or a <code>tidyselect</code> helper. Default is <code>dplyr::everything()</code> .

Details

This function makes it easy to extract a multiple instrument's data from a REDCapTidieR supertibble into a named list. Specifying instruments using `tidyselect` helper functions such as `dplyr::starts_with()` or `dplyr::ends_with()` is supported.

Value

A named list of tibbles

Examples

```
superheroes_supertbl

# Extract all data tibbles
extract_tibbles(superheroes_supertbl)

# Only extract data tibbles starting with "heroes"
extract_tibbles(superheroes_supertbl, starts_with("heroes"))
```

Description

Use these functions with the `format_labels` argument of `make_labelled()` to define how variable labels should be formatted before being applied to the data columns of `redcap_data`. These functions are helpful to create pretty variable labels from REDCap field labels.

- `fmt_strip_whitespace()` removes extra white space inside and at the start and end of a string. It is a thin wrapper of `stringr::str_trim()` and `stringr::str_squish()`.
- `fmt_strip_trailing_colon()` removes a colon character at the end of a string.
- `fmt_strip_trailing_punct()` removes punctuation at the end of a string.
- `fmt_strip_html()` removes html tags from a string.
- `fmt_strip_field_embedding()` removes text between curly braces `{}` which REDCap uses for special "field embedding" logic. Note that `read_redcap()` removes html tags and field embedding logic from field labels in the metadata by default.

Usage

```
fmt_strip_whitespace(x)
```

```
fmt_strip_trailing_colon(x)
```

```
fmt_strip_trailing_punct(x)
```

```
fmt_strip_html(x)
```

```
fmt_strip_field_embedding(x)
```

Arguments

`x` a character vector

Value

a modified character vector

Examples

```
fmt_strip_whitespace("Poorly Spaced Label ")
```

```
fmt_strip_trailing_colon("Label:")
```

```
fmt_strip_trailing_punct("Label-")
```

```
fmt_strip_html("<b>Bold Label</b>")
```

```
fmt_strip_field_embedding("Label{another_field}")

superheroes_supertbl

make_labelled(superheroes_supertbl, format_labels = fmt_strip_trailing_colon)
```

make_labelled	<i>Apply variable labels to a REDCapTidieR supertibble</i>
---------------	--

Description

Take a supertibble and use the labelled package to apply variable labels to the columns of the supertibble as well as to each tibble in the redcap_data, redcap_metadata, and redcap_events columns of that supertibble.

Usage

```
make_labelled(supertbl, format_labels = NULL)
```

Arguments

supertbl	a supertibble generated using read_redcap()
format_labels	one or multiple optional label formatting functions. A label formatting function is a function that takes a character vector and returns a modified character vector of the same length. This function is applied to field labels before attaching them to variables. One of: • NULL to apply no additional formatting. Default. • A label formatting function. • A character with the name of a label formatting function. • A vector or list of label formatting functions or function names to be applied in order. Note that ordering may affect results.

Details

The variable labels for the data tibbles are derived from the field_label column of the metadata tibble.

Value

A labelled supertibble.

Examples

```

superheroes_supertbl

make_labelled(superheroes_supertbl)

make_labelled(superheroes_supertbl, format_labels = tolower)

## Not run:
redcap_uri <- Sys.getenv("REDCAP_URI")
token <- Sys.getenv("REDCAP_TOKEN")

supertbl <- read_redcap(redcap_uri, token)
make_labelled(supertbl)

## End(Not run)

```

read_redcap

Import a REDCap database into a tidy supertibble

Description

Query the REDCap API to retrieve data and metadata about a project, and transform the output into a "supertibble" that contains data and metadata organized into tibbles, broken down by instrument.

Usage

```

read_redcap(
  redcap_uri,
  token,
  raw_or_label = "label",
  forms = NULL,
  export_survey_fields = NULL,
  export_data_access_groups = NULL,
  suppress_redcap_messages = TRUE,
  guess_max = Inf,
  allow_mixed_structure = getOption("redcaptidier.allow.mixed.structure", FALSE)
)

```

Arguments

redcap_uri	The URI/URL of the REDCap server (e.g., "https://server.org/apps/redcap/api/"). Required.
token	The user-specific string that serves as the password for a project. Required.
raw_or_label	A string (either 'raw', 'label', or 'haven') that specifies whether to export the raw coded values or the labels for the options of categorical fields. Default is 'label'. If 'haven' is supplied, categorical fields are converted to haven_labelled vectors.

forms	A character vector of REDCap instrument names that specifies which instruments to import. Default is NULL which imports all instruments in the project.
export_survey_fields	A logical that specifies whether to export survey identifier and timestamp fields. The default, NULL, tries to determine if survey fields exist and returns them if available.
export_data_access_groups	A logical that specifies whether to export the data access group field. The default, NULL, tries to determine if a data access group field exists and returns it if available.
suppress_redcap_messages	A logical to control whether to suppress messages from REDCapR API calls. Default TRUE.
guess_max	A positive <code>base::numeric</code> value passed to <code>readr::read_csv()</code> that specifies the maximum number of records to use for guessing column types. Default Inf.
allow_mixed_structure	A logical to allow for support of mixed repeating/non-repeating instruments. Setting to TRUE will treat the mixed instrument's non-repeating versions as repeating instruments with a single instance. Applies to longitudinal projects only. Default FALSE. Can be set globally with <code>options(redcaptidier.allow.mixed.structure = TRUE)</code> .

Details

This function uses the **REDCapR** package to query the REDCap API. The REDCap API returns a **block matrix** that mashes data from all data collection instruments together. The `read_redcap()` function deconstructs the block matrix and splices the data into individual tibbles, where one tibble represents the data from one instrument.

Value

A tibble in which each row represents a REDCap instrument. It contains the following columns:

- `redcap_form_name`, the name of the instrument
- `redcap_form_label`, the label for the instrument
- `redcap_data`, a tibble with the data for the instrument
- `redcap_metadata`, a tibble of data dictionary entries for each field in the instrument
- `redcap_events`, a tibble with information about the arms and longitudinal events represented in the instrument. Only if the project has longitudinal events enabled
- `structure`, the instrument structure, either "repeating" or "nonrepeating"
- `data_rows`, the number of rows in the instrument's data tibble
- `data_cols`, the number of columns in the instrument's data tibble
- `data_size`, the size in memory of the instrument's data tibble computed by `lobstr::obj_size()`
- `data_na_pct`, the percentage of cells in the instrument's data columns that are NA excluding identifier and form completion columns

Examples

```
## Not run:
redcap_uri <- Sys.getenv("REDCAP_URI")
token <- Sys.getenv("REDCAP_TOKEN")

read_redcap(
  redcap_uri,
  token,
  raw_or_label = "label"
)

## End(Not run)
```

superheroes_supertbl *Superheroes Data*

Description

A dataset of superheroes in a REDCapTidieR supertbl object

Usage

```
superheroes_supertbl
```

Format

```
heroes_information:
A tibble with 734 rows and 12 columns:
record_id REDCap record ID
name Hero name
gender Gender
eye_color Eye color
race Race
hair_color Hair color
height Height
weight Weight
publisher Publisher
skin_color Skin color
alignment Alignment
form_status_complete REDCap instrument completed?

super_hero_powers:
A tibble with 5,966 rows and 4 columns:
record_id REDCap record ID
redcap_form_instance REDCap repeat instance
power Super power
form_status_complete REDCap instrument completed?
```

Source

<https://www.superherodb.com/>

tbl_sum.redcap_supertbl

Provide a succinct summary of an object

Description

tbl_sum() gives a brief textual description of a table-like object, which should include the dimensions and the data source in the first element, and additional information in the other elements (such as grouping for **dplyr**). The default implementation forwards to [obj_sum\(\)](#).

Usage

```
## S3 method for class 'redcap_supertbl'
tbl_sum(x)
```

Arguments

x Object to summarise.

Value

A named character vector, describing the dimensions in the first element and the data source in the name of the first element.

vec_ptype_abbr.redcap_supertbl

Vector type as a string

Description

vec_ptype_full() displays the full type of the vector. vec_ptype_abbr() provides an abbreviated summary suitable for use in a column heading.

Usage

```
## S3 method for class 'redcap_supertbl'
vec_ptype_abbr(x, ..., prefix_named, suffix_shape)
```

Arguments

x A vector.
... These dots are for future extensions and must be empty.
prefix_named If TRUE, add a prefix for named vectors.
suffix_shape If TRUE (the default), append the shape of the vector.

Value

A string.

write_redcap_xlsx	<i>Write Supertibbles to XLSX</i>
-------------------	-----------------------------------

Description

Transform a supertibble into an XLSX file, with each REDCap data tibble in a separate sheet.

Usage

```
write_redcap_xlsx(
  supertbl,
  file,
  add_labelled_column_headers = NULL,
  use_labels_for_sheet_names = TRUE,
  include_toc_sheet = TRUE,
  include_metadata_sheet = TRUE,
  table_style = "tableStyleLight8",
  column_width = "auto",
  recode_logical = TRUE,
  na_replace = "",
  overwrite = FALSE
)
```

Arguments

supertbl	A supertibble generated using read_redcap() .
file	The name of the file to which the output will be written.
add_labelled_column_headers	If TRUE, the first row of each sheet will contain variable labels, with variable names in the second row. If FALSE, variable names will be in the first row. The default value, NULL, tries to determine if supertbl contains variable labels and, if present, includes them in the first row. The labelled package must be installed if add_labelled_column_headers is TRUE.
use_labels_for_sheet_names	If FALSE, sheet names will come from the REDCap instrument names. If TRUE, sheet names will come from instrument labels. The default is TRUE.
include_toc_sheet	If TRUE, the first sheet in the XLSX output will be a table of contents, providing information about each data tibble in the workbook. The default is TRUE.
include_metadata_sheet	If TRUE, the final sheet in the XLSX output will contain metadata about each variable, combining the content of supertbl\$redcap_metadata. The default is TRUE.

table_style	Any Excel table style name or "none". For more details, see the "formatting" vignette of the openxlsx package. The default is "tableStyleLight8".
column_width	Sets the width of columns throughout the workbook. The default is "auto", but you can specify a numeric value.
recode_logical	If TRUE, fields with "yesno" field type are recoded to "yes"/"no" and fields with a "checkbox" field type are recoded to "Checked"/"Unchecked". The default is TRUE.
na_replace	The value used to replace NA values in supertbl. The default is "".
overwrite	If FALSE, will not overwrite file when it exists. The default is FALSE.

Value

An openxlsx2 workbook object, invisibly

Examples

```
## Not run:
redcap_uri <- Sys.getenv("REDCAP_URI")
token <- Sys.getenv("REDCAP_TOKEN")

supertbl <- read_redcap(redcap_uri, token)

supertbl %>%
  write_redcap_xlsx(file = "supertibble.xlsx")

# Add variable labels

library(labelled)

supertbl %>%
  make_labelled() %>%
  write_redcap_xlsx(file = "supertibble.xlsx", add_labelled_column_headers = TRUE)

## End(Not run)
```

Index

- * **datasets**
 - superheroes_supertbl, [12](#)
- add_skimr_metadata, [2](#)
- base::numeric, [11](#)
- bind_tibbles, [3](#)
- combine_checkboxes, [4](#)
- combine_checkboxes(), [4](#), [5](#)
- extract_tibble, [6](#)
- extract_tibbles, [7](#)
- fmt_strip_field_embedding
(format_helpers), [8](#)
- fmt_strip_html (format_helpers), [8](#)
- fmt_strip_trailing_colon
(format_helpers), [8](#)
- fmt_strip_trailing_punct
(format_helpers), [8](#)
- fmt_strip_whitespace (format_helpers), [8](#)
- format_helpers, [8](#)
- get_default_skimmer_names, [2](#)
- make_labelled, [9](#)
- obj_sum(), [13](#)
- read_redcap, [10](#)
- read_redcap(), [4](#), [5](#), [14](#)
- readr::read_csv(), [11](#)
- skim, [2](#)
- skimr, [2](#)
- superheroes_supertbl, [12](#)
- tbl_sum.redcap_supertbl, [13](#)
- vctrs::vec_as_names(), [4](#)
- vec_ptype_abbr.redcap_supertbl, [13](#)
- write_redcap_xlsx, [14](#)