

Package ‘REDCapTidieR’

November 8, 2023

Type Package

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

Version 1.0.0

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, lobstr, lubridate, purrr, REDCapR (>= 1.1.0), rlang, stringi, stringr, tibble, tidyr, tidyselect, formattable

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

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

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

Maintainer Richard Hanna <richardshanna91@gmail.com>

Repository CRAN

Date/Publication 2023-11-08 18:20:09 UTC

R topics documented:

add_skimr_metadata	2
bind_tibbles	3
extract_tibble	4
extract_tibbles	4
format-helpers	5
make_labelled	6
read_redcap	7
superheroes_supertbl	9
write_redcap_xlsx	10
Index	12

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

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)
```

extract_tibble	<i>Extract a single data tibble from a REDCapTidieR supertibble</i>
----------------	---

Description

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

Usage

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

Arguments

supertbl	A supertibble generated by <code>read_redcap()</code> . Required.
tbl	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

supertbl	A supertibble generated by <code>read_redcap()</code> . Required.
tbls	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"))
```

format-helpers

Format REDCap variable labels

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 = .Machine$integer.max
)
```

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' or 'label') that specifies whether to export the raw coded values or the labels for the options of categorical fields. Default is 'label'.
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 <code>.Machine\$integer.max</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 super_tbl 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/>

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, 9
- add_skimr_metadata, 2
- base::numeric, 8
- bind_tibbles, 3
- extract_tibble, 4
- extract_tibbles, 4
- fmt_strip_field_embedding
 - (format-helpers), 5
- fmt_strip_html (format-helpers), 5
- fmt_strip_trailing_colon
 - (format-helpers), 5
- fmt_strip_trailing_punct
 - (format-helpers), 5
- fmt_strip_whitespace (format-helpers), 5
- format-helpers, 5
- get_default_skimmer_names, 2
- make_labelled, 6
- read_redcap, 7
- readr::read_csv(), 8
- skim, 2
- skimr, 2
- superheroes_supertbl, 9
- write_redcap_xlsx, 10