Package: redcaptools (via r-universe)

October 21, 2024

Type Package

Title Tools for exporting and working with REDCap data

Version 0.4.0

Maintainer Alan G Haynes <alan.haynes@unibe.ch>

Description Tools for exporting and working with REDCap data (e.g. adding labels, formatting dates).

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.0

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Imports crayon, dplyr, httr2 (>= 0.2.0), labelled, lubridate, magrittr, stringr, tidyr

Config/testthat/edition 3

Repository https://ctu-bern.r-universe.dev

RemoteUrl https://github.com/CTU-Bern/redcaptools

RemoteRef HEAD

RemoteSha 74dcc61efafb6162f0ae59c40303a07354ce91ab

Contents

`convert_to_logical` *Convert variables to logical*

Description

This is particularly useful for binary variables that have been encoded with e.g. Yes and No as options. Variable labels are retained, which may or may not make sense, depending on the variable

Usage

```
convert_to_logical(
  data,
  vars,
  true = "Yes",
  replace = TRUE,
  append = "_logical"
)
```

Arguments

`data`	dataframe
`vars`	character string of variables to convert
`true`	value which should become `TRUE`
`replace`	Replace the indicated variables
`append`	text to append to new variables (when `replace = TRUE`)

Value

`data` with modified variables, potentially with additional variables (if `replace = TRUE`)

Examples

```
data(mtcars)
convert_to_logical(mtcars, "am", 1)
convert_to_logical(mtcars, c("am", "vs"), 1)
convert_to_logical(mtcars, c("am", "vs"), 1, FALSE)
convert_to_logical(mtcars, c("am", "vs"), 1, FALSE, "_lgl")
```

`deprecated` *Deprecated functions These functions have been renamed to be more consistent with the rest of the package. They may be removed in a future version.*

Description

Deprecated functions These functions have been renamed to be more consistent with the rest of the package. They may be removed in a future version.

Usage

```
rc_prep(
  data,
  metadata,
  rep = FALSE,
  rep_date = rep,
  rep_datetime = rep,
  rep_singlechoice = rep,
  rep_multichoice = rep,
  app_date = "_date",
  app_datetime = "_datetime",
  app_singlechoice = "_factor",
  app_multichoice = "_factor",
  ...
)

rc_dates(data, metadata, replace = FALSE, append = "_date")

rc_datetimes(data, metadata, replace = FALSE, append = "_datetime", ...)

split_by_form(data, metadata)
```

Arguments

`data`	dataframe
`metadata`	datadictionary as exported from REDCap or downloaded from the API
`rep`	replace variables. If FALSE, encoded versions of the variable will be created
`rep_date`, `rep_datetime`, `rep_singlechoice`, `rep_multichoice`	replace the indicated variable type

`app_date`, `app_datetime`, `app_singlechoice`, `app_multichoice`	text to append to the newly generated variables name (if `rep_*` is FALSE)
`...`	options passed to/from other methods
`replace`	indicator of whether to replace original variables or not
`append`	text to append to the newly generated variables name (if `replace` is TRUE)

Value

list of dataframes

Functions

- `rc_prep()`: original function name for `redcap_prep`
- `rc_dates()`: original function name for `redcap_dates`
- `rc_datetimes()`: original function name for `redcap_datetimes`
- `split_by_form()`: deprecated in favour of `redcap_toform` Split a manually exported REDCap dataset into forms

---

`importdemo_data` *Example import data*

Description

Simulated import data for the most common REDCap field types and validations to be used with the data dictionary `importdemo_dict` for testing of `redcap_import_select` and `redcap_import_recode`. Item names and coding show mismatches on purpose so that the interactive functions can be tested.

Usage

```
importdemo_data
```

Format

A data frame with 17 observations and 32 variables.

`importdemo_dict` *Example data dictionary*

Description

Simulated data dictionary for the most common REDCap field types and validations to be used with the sample data `importdemo_data` for testing of `redcap_import_select` and `redcap_import_recode`. Item names and coding show mismatches on purpose so that the interactive functions can be tested.

Usage

```
importdemo_dict
```

Format

A data frame with 38 variable specifications.

`label_others` *Label non-single/multiple choice/date(time) fields* `singlechoice_factor`*,* `multichoice_factor`*,* `rc_date` *and* `rc_datetime`

Description

Label non-single/multiple choice/date(time) fields `singlechoice_factor`, `multichoice_factor`, `rc_date` and `rc_datetime`

Usage

```
label_others(data, metadata)
```

Arguments

`data`	dataframe
`metadata`	redcap data dictionary

multichoice_factor *create factors for multiple choice variables*

Description

Converts the numeric values returned from REDCap to factors (with levels Yes/No). This function also applies labels to the variable itself, based on the option label.

Usage

```
multichoice_factor(
  data,
  metadata,
  replace = FALSE,
  append = "_factor",
  include_vlabel = FALSE,
  vlabel_sep = ": "
)
```

Arguments

`data`	the data.frame to modify
`metadata`	metadata/datadictionary
`replace`	whether to overwrite the existing data .
`append`	text to append to the variable name if not overwriting
`include_vlabel`	logical indicating whether to include the variable label before the value label
`vlabel_sep`	text to use for separating vlabel and label

Value

input data.frame with additional factor variables.

redcap_dates *REDCap Date Conversion*

Description

This function is used to prepare date fields in a data table for import in REDCap. The function tries to take all (most)possibilities to enter dates into account and converts them to as.Date format.

Usage

```
redcap_dates(var, unk_day = "01", unk_month = "01", unk_cent = "20")
```

Arguments

`var`	variable to convert
`unk_day`	Day to use if unknown, i.e. if only the year or only the month + year is found. The default is 01 (2022 -> 2022-01-01).
`unk_month`	Month to use if unknown, i.e. if only the year is found. The default is 01 (2022 -> 2022-01-01).
`unk_cent`	Century to use if unknown, i.e. if only the year is found. The default is 20 (22 -> 2022)

Value

converted variable

Examples

```
var <-c("01.12.2022", "12.2022", "2022", "01/12/2022")
redcap_dates(var)
```

`redcap_export_batch` *Export data in batches*

Description

Exports of large databases may fail using the standard export methods implemented in redcap_export_tbl and redcap_export_byform. To remedy this, the `redcap_export_batch` function exports data in smaller chunks (of 1000 records by default)

Usage

```
redcap_export_batch(
  token,
  url,
  batchsize = 1000,
  meta = NULL,
  byform = FALSE,
  remove_empty = TRUE,
  ...
)
```

Arguments

`token`	REDcap API token
`url`	address of the API
`batchsize`	number of records per batch
`meta`	metadata from `redcap_export_meta` (will be downloaded if not provided)

`byform`	logical. Download data by form (see redcap_export_byform)
`remove_empty`	when using byform: should empty rows be removed from the dataset (REDCap automatically creates all forms for an event when any form in the event is created)
`...`	other parameters passed to the API (see your REDCap API documentation for options)

Value

depending on `byform`, either a list of dataframes or a single dataframe

See Also

redcap_export_tbl, redcap_export_byform

Examples

```
# token <- "some_really_long_string_provided_by_REDCap"
# as a single dataframe
# redcap_export_batch(token, "https://www.some_redcap_url.com/api/")
# as a list of dataframes (forms)
# redcap_export_batch(token, "https://www.some_redcap_url.com/api/", byform = TRUE)
```

`redcap_export_byform` *Export REDCap data by form*

Description

Export REDCap data by form

Usage

```
redcap_export_byform(
  token,
  url,
  meta = NULL,
  remove_empty = TRUE,
  wait = 0.2,
  ...
)
```

Arguments

`token`	REDcap API token
`url`	address of the API
`meta`	metadata from `redcap_export_meta` (will be downloaded if not provided)

`remove_empty`	should empty rows be removed from the dataset (REDCap automatically creates all forms for an event when any form in the event is created)
`wait`	seconds to wait between API calls
`...`	other parameters passed to the API (see your REDCap API documentation for options)

Value

list of dataframes

Examples

```
# token <- "some_really_long_string_provided_by_REDCap"
# redcap_export_byform(token, "https://www.some_redcap_url.com/api/")
```

`redcap_export_meta` *Export the most important REDCap metadata tables*

Description

The REDCap API has a large number of API endpoints. Those that are metadata-type details are listed on this page. The

Usage

```
redcap_export_meta(
  token,
  url,
  tabs = c("metadata", "event", "formEventMapping", "instrument"),
  ...
)
```

Arguments

`token`	REDcap API token
`url`	address of the API
`tabs`	tables to export. `project` is always added.
`...`	other parameters passed to the API (see your REDCap API documentation for options)

Details

Allowed tabs are

- `arm` - labels of a projects arms
- `dag` - data access groups (DAGs)
- `userDagMapping` - mapping between users and DAGs
- `event` - list of events in the project (only available for longitudinal projects)
- `exportFieldNames` - list of the fields that the API returns
- `instrument` - list of instruments (eCRFs/forms) in the project
- `formEventMapping` - mapping between instruments (forms) and events (only available for longitudinal projects)
- `metadata` - the data dictionary
- `project` - information on the project
- `record` - the data itself. The method has many options. See the API help page on your REDCap instance
- `repeatingFormsEvents` - which forms can repeat on which events
- `report` - access custom reports defined in REDCap
- `version` - REDCap version
- `user` - list of users
- `userRole` - rights for each role
- `userRoleMapping` - user-roll mapping

Value

list of dataframes

Note

tables that are not relevant for non-longitudinal projects (e.g. formEventMapping and event) are silently removed

Examples

```
# token <- "some_really_long_string_provided_by_REDCap"
# redcap_export_meta(token, "https://www.some_redcap_url.com/api/")
```

`redcap_export_tbl` *Export tables from REDCap*

Description

Export tables from REDCap

Usage

```
redcap_export_tbl(token, url, content, ...)
```

Arguments

`token`	REDcap API token
`url`	address of the API
`content`	content to download
`...`	other parameters passed to the API (see your REDCap API documentation for options)

Value

dataframe

Examples

```
# token <- "some_really_long_string_provided_by_REDCap"
# redcap_export_tbl(token, "https://www.some_redcap_url.com/api/", "record")
```

`redcap_import_recode` *REDCap Recode*

Description

This function loops through all the variable classes of a data set and lets the user compare them with the variable types set up in REDCap. An API token is needed to download the variable names from REDCap. The class/type can be changed to output a new summary and to match REDCap. If 'factor' is chosen as class, the script loops through all the factor levels and compares them with the coding as defined in REDCap. The factor levels can then be matched with the respective codes in REDCap. The function returns a data frame with the recoded variables and writes the executed code to a log-file for copy-pasting and adjusting/reusing.

Usage

```
redcap_import_recode(selected_data, dict = NULL, rc_token, rc_url)
```

Arguments

`selected_data`	Data to be recoded
`dict`	Data dictionary (e.g. as downloaded from REDCap or via `redcap_export_meta(rc_token, rc_url)$meta`). If not supplied, this will be downloaded from the API using `rc_token`.
`rc_token`	REDCap API token
`rc_url`	Link to REDCap API. Default: https://redcap.ctu.unibe.ch/api/

Value

Data frame with recoded data. Log-file with executed code.

Examples

```
# data(importdemo_data)
# data(importdemo_dict)
# redcap_import_recode(importdemo_data, importdemo_dict)

# if using local data:
# token <- "xxxxx"
# url <- "xxxxx"
# file <- "data.csv"
# redcap_import_recode(file, rc_token = token, rc_url = url)
```

`redcap_import_select` *REDCap Select and Rename*

Description

This function loops through all the variable names of a data set and lets the user compare them with the variable names set up in REDCap.
The REDCap data dictionary can either be directly provided or downloaded from the REDCap project by providing an API token and matching URL.
For variables with matching names in REDCap, the user can decide to automatically select them without renaming. If auto-selecting is turned off, the user can decide to not select these variables at all or to select and rename them.
For variables without matching names in REDCap, the user will always be prompted to decide either to not select these variables at all or to select and rename them.
The function returns a data frame with the selected/renamed variables, writes an overview csv-table, and a short summary with the executed code to a log-file for copy-pasting and adjusting/reusing.

Usage

```
redcap_import_select(
  import_data,
  dict = NULL,
  rc_token,
  rc_url,
  forms = NULL,
  auto_match = TRUE,
  auto_skip_nomatch = FALSE,
  skip_intro = FALSE,
  suppress_txt = FALSE,
  log = TRUE,
  log_code = "redcap_import_select_code.txt",
  log_table = "redcap_import_select_overview.txt",
  wait = 2
)
```

Arguments

`import_data`	Data frame to be imported
`dict`	Data dictionary (e.g. as downloaded from REDCap or via `redcap_export_meta(rc_token,rc_url)$me` If not supplied, this will be downloaded from the API using `rc_token` and `rc_url`.
`rc_token`	REDCap API token
`rc_url`	Link to REDCap API
`forms`	Character vector of the forms as set up in REDCap of which variable names will be displayed. Default = all forms.
`auto_match`	If TRUE, variables with matching names will be automatically selected. If FALSE, the user can decide if the variable shall be imported or not. Default = TRUE.
`auto_skip_nomatch`	If TRUE, variables without matching names will be automatically skipped. If FALSE, the user can decide to select and rename the variable. Default = FALSE.
`skip_intro`	If set to TRUE, the introduction messages will be skipped. Default = FALSE
`suppress_txt`	If set TRUE, all text output will be suppressed (not recommended). Default = FALSE.
`log`	If TRUE, an overview csv-table, and a log-file are stored in the working directory. Default = TRUE.
`log_code`	Name and location of the log-file containing the executed code. Default = redcap_import_select_code.txt.
`log_table`	Name and location of the csv.table containing the tabular overview. Default = redcap_import_select_overview.csv.
`wait`	Allows you to set the latency time between the steps. Default = 2s.

Value

Data frame with selected/renamed data. Log-file with executed code. CSV-table with overview.

Examples

```
# data(importdemo_data)
# data(importdemo_dict)
# redcap_import_select(importdemo_data, importdemo_dict)

# if using local data:
# token <- "xxxxx"
# url <- "xxxxx"
# file <- "data.csv"
# redcap_import_select(file, rc_token = token, rc_url = url)
```

---

redcap_prep — *Convert REDCap variable types (dates, datetimes, factors) and apply labels*

Description

Convert REDCap variable types (dates, datetimes, factors) and apply labels

Usage

```
redcap_prep(
  data,
  metadata,
  rep = FALSE,
  rep_date = rep,
  rep_datetime = rep,
  rep_singlechoice = rep,
  rep_multichoice = rep,
  app_date = "_date",
  app_datetime = "_datetime",
  app_singlechoice = "_factor",
  app_multichoice = "_factor",
  ...
)
```

Arguments

`data`	dataframe
`metadata`	data dictionary from REDCap
`rep`	replace variables. If FALSE, encoded versions of the variable will be created
`rep_date`, `rep_datetime`, `rep_singlechoice`, `rep_multichoice`	replace the indicated variable type
`app_date`, `app_datetime`, `app_singlechoice`, `app_multichoice`	text to append to the newly generated variables name (if `rep_*` is FALSE)
`...`	options passed to/from other methods

Value

dataframe with converted factors, dates, POSIX, ...

`redcap_prep_dates` *Convert dates stored as strings to* `Date` *variables*

Description

Converts the string values returned from REDCap to Dates. This function also applies labels to the variable itself, based on the option label.

Usage

```
redcap_prep_dates(data, metadata, replace = FALSE, append = "_date")

redcap_prep_datetimes(
  data,
  metadata,
  replace = FALSE,
  append = "_datetime",
  ...
)
```

Arguments

`data`	the data.frame to modify
`metadata`	metadata/datadictionary
`replace`	whether to overwrite the existing data .
`append`	text to append to the variable name if not overwriting
`...`	options passed to/from other methods

Value

input data.frame with additional date variables/variables converted to dates.

Functions

- `redcap_prep_datetimes()`: input data.frame with date-time variables reformated to POSIX `redcap_toform` *Convert manually downloaded REDCap data into a list of forms*

Description

Similar to `redcap_export_byform`, this function tries to split a manually downloaded dataset into it's constituent forms. While use of the API allows individual forms to be downloaded, with a manual download, only the data dictionary is available as auxillary information. If no data dictionary is available, the function will use the variable names to guess the forms (see details).

Usage

```
redcap_toform(data, datadict = NULL, metadata = NULL, guess_events = TRUE, ...)
```

Arguments

`data`	imported REDCap data
`datadict`	data dictionary downloaded manually from REDCap
`metadata`	metadata downloaded from REDCap API
`guess_events`	restrict forms to events (rows) where data exists (see details)
`...`	additional arguments passed to other functions (currently unused)

Details

In a longitudinal data collection with many forms, a REDCap dataset will have a large degree of empty cells. The `guess_events` argument uses missingness as an indicator of a row not being part of the form in question. If all user variables (i.e. those that do not start with `redcap`) are empty, the row will be removed from the dataset.

If neither `datadict` nor `metadata` are provided, the function will attempt to guess the forms based on the variable names, specifically the `form_complete` variables which denote the state of the form. This is not a foolproof method: there may be other variables in the data that end with `_complete`.

Examples

```
data <- readRDS(system.file("extdata/test.rda", package = "redcaptools"))
metadata <- readRDS(system.file("extdata/meta.rda", package = "redcaptools"))
dd <- read.csv(system.file("extdata/DataDictionary.csv", package = "redcaptools"))
redcap_toform(data, dd)
redcap_toform(data, metadata = metadata)
redcap_toform(data)
```

`remove_empty_rows` *Analagous to ‘janitor::remove_empty(..., "rows")‘, but allows ignoring specific variables*

Description

Analagous to ‘janitor::remove_empty(..., "rows")‘, but allows ignoring specific variables

Usage

```
remove_empty_rows(data, ignore = "^(record_id|redcap)|_complete$")
```

Arguments

`data`	a dataframe
`ignore`	regex identifying variables to ignore

Value

dataframe

Examples

```
x <- data.frame(a = c(1:9, NA), b = rep(c("b", NA), 5))
remove_empty_rows(x, "a")
remove_empty_rows(x, FALSE)
```

`singlechoice_factor` *create factors for single choice variables*

Description

Converts the numeric values returned from REDCap to factors. This function also applies labels to the variable itself.

Usage

```
singlechoice_factor(data, metadata, replace = FALSE, append = "_factor")
```

Arguments

`data`	the data.frame to modify
`metadata`	metadata/datadictionary
`replace`	whether to overwrite the existing data .
`append`	text to append to the variable name if not overwriting

Value

dataframe with factor variables

---

singlechoice_opts *Get options for single and multi choice questions*

Description

Get options for single and multi choice questions

Usage

```
singlechoice_opts(metadata)

multichoice_opts(metadata)
```

Arguments

`metadata`	data.frame containing the metadata

Details

Multiple choice variables exist in REDCap data as a set of 0/1/TRUE/FALSE variables, where 1/TRUE represents a selected/checked answer. Hence, for a single multiple choice 'question' in the datadictionary/metadata with `n` options, there are `n` variables. Each variable is the variable name (e.g. morbidities) followed by 3 underscores (`___`) and the option number (e.g. 1) - `morbidities___1`.

Value

data.frame with variables `var` (variable), `label` (the variable label), `vals` (possible values for the variable) and `labs` (the labels related to each value in `vals`)

data.frame with variables `ovar` (the variable as it appears in the data dictionary/metadata), `var` (the variable as it appears in the data itself), `vlabel` (the variable label), `vals` (possible values for the variable) and `labs` (the labels related to each value in `vals`)

Index