Package 'REDCapR'

October 4, 2019

Title Interaction Between R and REDCap

Description Encapsulates functions to streamline calls from R to the REDCap API. REDCap (Research Electronic Data CAPture) is a web application for building and managing online surveys and databases developed at Vanderbilt University. The Application Programming Interface (API) offers an avenue to access and modify data programmatically, improving the capacity for literate and reproducible programming.

Version 0.10.2.9001

Date 2019-10-04

URL https://github.com/OuhscBbmc/REDCapR, http://ouhsc.edu/bbmc/, http://project-redcap.org

BugReports https://github.com/OuhscBbmc/REDCapR/issues

Depends R(¿= 3.0.0),
stats

Imports checkmate (¿= 1.8.4),
dplyr (¿= 0.5.0),
httr (¿= 1.3.0),
magrittr,
methods,
readr (¿= 1.3.1),
rlang,
tibble (¿= 1.4.0),
tidyr (¿= 0.7.0)

Suggests DBI (¿= 0.7.0),
devtools (¿= 1.13.0),
kableExtra,
knitr (¿= 1.18.0),
odbc (¿= 1.1.1),
rmarkdown,
sessioninfo,
testthat (¿= 0.9)

License GPL-2

LazyData TRUE

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 6.1.1

Roxygen list(markdown = TRUE)

R topics documented:

`REDCapR-package` *R utilities for interacting with a REDCap data system* `http://www.project-redcap.org/`

Description

Comprehensive documentation is also available at https://ouhscbbmc.github.io/REDCapR/.

Much of this package has been developed to support the needs of the following projects. We appreciate the support.

- *OUHSC CCAN Independent Evaluation of the State of Oklahoma Competitive Maternal, Infant, and Early Childhood Home Visiting (MIECHV) Project.* HRSA/ACF D89MC23154. David Bard, PI, OUHSC; 2011-2015.
- *Independent Evaluation of the State of OK MIECHV Evidence Based Home Visitation Project*, NIH-sponsored collaboration with OSDH. David Bard, PI, OUHSC; 2015-2017.
- *OSDH ParentPRO Pilot Evaluation*, federally-sponsored collaboration with OSDH. David Bard, PI, OUHSC; 2015-2017.
- *Title IV-E Waiver Project*, HRSA/MCHB-sponsored collaboration with OKDHS; David Bard, PI, OUHSC; 2014-2017.
- *Integrative Analysis of Longitudinal Studies of Aging (IALSA)*, sponsored by NIH 5P01AG043362. Scott Hofer, PI, University of Victoria; Will Beasley, PI of site-award, OUHSC; 2013-2018.
- *Oklahoma Shared Clinical and Translational Resources*, sponsored by NIH NIGMS; U54 GM104938. Judith A. James, PI, OUHSC; 2013-2018.
- Additional Institutional Support from OUHSC Dept of Pediatrics; 2013-2017.

Note

The release version is available through CRAN by running `install.packages('REDCapR')`. The most recent development version is available through GitHub by running `devtools::install_github('OuhscBb` (make sure devtools is already installed). If you're having trouble with the package, please install the development version. If this doesn't solve your problem, please create a new issue, or email Will.

See REDCapR's advanced vignette for information and examples for overriding the default SSL options.

Examples

```
## Not run:
# Install/update REDCapR with the release version from CRAN.
install.packages('REDCapR')

# Install/update REDCapR with the development version from GitHub
#install.packages('devtools') #Uncomment if `devtools` isn't installed already.
devtools::install_github('OuhscBbmc/REDCapR')

## End(Not run)
```

`collapse_vector` *Collapse a vector of values into a single string when necessary.*

Description

REDCap's API frequently specifies a series of values separated by commas. In the R world, it's easier to keep these values as separate elements in a vector. This functions squashes them together in a single character element (presumably right before the return value is passed to the API)

Usage

```
collapse_vector(elements, collapsed)
```

Arguments

elements	An array of values. Can be NULL. Required.
collapsed	A single character element, where the values are separated by commas. Can be NULL. Required.

Value

A single character element, where the values are separated by commas. Can be blank. (*i.e.*, "").

Author(s)

Will Beasley

Examples

```
library(REDCapR) #Load the package into the current R session.
REDCapR:::collapse_vector(elements=NULL, collapsed=NULL)
REDCapR:::collapse_vector(elements=letters, collapsed=NULL)
REDCapR:::collapse_vector(elements=NULL, collapsed="4,5,6")
```

---

constant *Collection of REDCap-specific constants*

Description

Collection of constants defined by the REDCap developers.

Usage

```
constant(name, simplify = TRUE)
```

Arguments

name	Name of constant. Required character.
simplify	Simplifies the vector of values to a common data-type, if possible. Passed to the simplify parameter of base::sapply().

Details

Form Completeness

The current constants relate to the 'complete' variable at the end of each form.

- `form_incomplete`: 0L (*i.e.*, an integer)
- `form_unverified`: 1L
- `form_complete`: 2L

Export Rights

See https://your-server/redcap/api/help/?content=exp_users.

- `data_export_rights_no_access` : 0L
- `data_export_rights_deidentified` : 1L
- `data_export_rights_full` : 2L

Form Rights

See https://your-server/redcap/api/help/?content=exp_users. The order of these digits may be unexpected.

- `form_rights_no_access` : 0L
- `form_rights_readonly` : 2L
- `form_rights_edit_form` : 1L
- `form_rights_edit_survey` : 3L

Access Rights

See https://your-server/redcap/api/help/?content=exp_users.

- `access_no` : 0L
- `access_yes` : 1L

To add more, please for and edit constant.R on GitHub and submit a pull request. For instructions, please see Editing files in another user's repository in the GitHub documentation.

Value

The constant's value. Currently all are single integers, but that could be expanded in the future.

Author(s)

Will Beasley

Examples

```
REDCapR::constant("form_incomplete")  # Returns 0L
REDCapR::constant("form_unverified")  # Returns 1L
REDCapR::constant("form_complete"  )  # Returns 2L

REDCapR::constant("data_export_rights_no_access"   )  # Returns 0L
REDCapR::constant("data_export_rights_deidentified")  # Returns 1L
REDCapR::constant("data_export_rights_full"        )  # Returns 2L

REDCapR::constant("form_rights_no_access")     # Returns 0L
REDCapR::constant("form_rights_readonly" )     # Returns 2L --Notice the order
REDCapR::constant("form_rights_edit_form")     # Returns 1L
REDCapR::constant("form_rights_edit_survey")   # Returns 3L

REDCapR::constant("access_no" )  # Returns 0L
REDCapR::constant("access_yes")  # Returns 1L

REDCapR::constant(c("form_complete", "form_complete", "form_incomplete")) # Returns c(2L, 2L, 0L)
REDCapR::constant(c(
  "form_rights_no_access",
  "form_rights_readonly",
  "form_rights_edit_form",
  "form_rights_edit_survey"
)) # Returns c(0L, 2L, 1L, 3L)


constant_to_form_completion( c(0, 2, 1, 2, NA))
constant_to_form_rights(     c(0, 2, 1, 2, NA))
constant_to_export_rights(   c(0, 2, 1, 3, NA))
constant_to_access(          c(0, 1, 1, 0, NA))

## Not run:
# The following line returns an error:
#     Assertion on 'name' failed: Must be a subset of
#     {'form_complete','form_incomplete','form_unverified'},
#     but is {'bad-name'}.

REDCapR::constant("bad-name")    # Returns an error

REDCapR::constant(c("form_complete", "bad-name")) # Returns an error

## End(Not run)
```

---

create_batch_glossary *Creates a* base::data.frame() *that help batching long-running read and writes*

Description

The function returns a base::data.frame() that other functions use to separate long-running read and write REDCap calls into multiple, smaller REDCap calls. The goal is to

(1) reduce the chance of time-outs, and (2) introduce little breaks between batches so that the server isn't continually tied up.

Usage

```
create_batch_glossary(row_count, batch_size)
```

Arguments

row_count	The number records in the large dataset, before it's split.
batch_size	The maximum number of subject records a single batch should contain.

Details

This function can also assist splitting and saving a large base::data.frame() to disk as smaller files (such as a .csv). The padded columns allow the OS to sort the batches/files in sequential order.

Value

Currently, a base::data.frame() is returned with the following columns,

- id: an integer that uniquely identifies the batch, starting at 1.
- start_index: the index of the first row in the batch. integer.
- stop_index: the index of the last row in the batch. integer.
- id_pretty: a character representation of id, but padded with zeros.
- start_index: a character representation of start_index, but padded with zeros.
- stop_index: a character representation of stop_index, but padded with zeros.
- label: a character concatenation of id_pretty, start_index, and stop_index_pretty.

Author(s)

Will Beasley

See Also

See redcap_read() for a function that uses create_batch_gloassary.

Examples

```
REDCapR::create_batch_glossary(100, 50)
REDCapR::create_batch_glossary(100, 25)
REDCapR::create_batch_glossary(100, 3)
d <- data.frame(
  record_id = 1:100,
  iv        = sample(x=4, size=100, replace=TRUE),
  dv        = rnorm(n=100)
)
REDCapR::create_batch_glossary(nrow(d), batch_size=40)
```

kernel_api *REDCapR internal function for calling the REDCap API*

Description

This function is used by other functions to read and write values.

Usage

```
kernel_api(redcap_uri, post_body, config_options)
```

Arguments

redcap_uri	The URI (uniform resource identifier) of the REDCap project. Required.
post_body	List of contents expected by the REDCap API. Required.
config_options	A list of options to pass to POST method in the httr package. See the details below. Optional.

Details

If the API call is unsuccessful, a value of base::package_version("0.0.0") will be returned. This ensures that a the function will always return an object of class base::package_version. It guarantees the value can always be used in utils::compareVersion().

Value

A utils::packageVersion.

Examples

```
config_options <- NULL
uri            <- "https://bbmc.ouhsc.edu/redcap/api/"
token          <- "9A81268476645C4E5F03428B8AC3AA7B"
post_body      <- list(
  token    = token,
  content  = 'project',
  format   = 'csv'
)
kernel <- REDCapR:::kernel_api(uri, post_body, config_options)

# Consume the results in a few different ways.
kernel$result
read.csv(text=kernel$raw_text, stringsAsFactors=FALSE)
as.list(read.csv(text=kernel$raw_text, stringsAsFactors=FALSE))
```

`metadata_utilities` *Manipulate and interpret the metadata of a REDCap project*

Description

A collection of functions that assists handling REDCap project metadata.

Usage

```
regex_named_captures(pattern, text, perl = TRUE)

checkbox_choices(select_choices)
```

Arguments

`pattern`	The regular expression pattern. Required.
`text`	The text to apply the regex against. Required.
`perl`	Indicates if perl-compatible regexps should be used. Default is `TRUE`. Optional.
`select_choices`	The text containing the choices that should be parsed to determine the `id` and `label` values. Required.

Details

The `regex_named_captures()` function is general, and not specific to REDCap; it accepts any arbitrary regular expression. It returns a `base::data.frame()` with as many columns as named matches.

The `checkbox_choices()` function is specialized, and accommodates the "select choices" for a *single* REDCap checkbox group (where multiple boxes can be selected). It returns a `base::data.frame()` with two columns, one for the numeric id and one for the text label.

Value

Currently, a `base::data.frame()` is returned a row for each match, and a column for each *named* group within a match. For the `retrieve_checkbox_choices()` function, the columns will be.

- `id`: The numeric value assigned to each choice (in the data dictionary).
- `label`: The label assigned to each choice (in the data dictionary).

Author(s)

Will Beasley

References

See the official documentation for permissible characters in a checkbox label. *I'm bluffing here, because I don't know where this is located. If you know, please tell me.*

Examples

```
#The weird ranges are to avoid the pipe character; PCRE doesn't support character negation.
pattern_boxes <- "(?<=\\A| \\| )(?<id>\\d{1,}), (?<label>[\x20-\x7B\x7D-\x7E]{1,})(?= \\| |\\Z)"

choices_1 <- paste0(
  "1, American Indian/Alaska Native | ",
  "2, Asian | ",
  "3, Native Hawaiian or Other Pacific Islander | ",
  "4, Black or African American | ",
  "5, White | ",
  "6, Unknown / Not Reported")

#This calls the general function, and requires the correct regex pattern.
REDCapR::regex_named_captures(pattern=pattern_boxes, text=choices_1)

#This function is designed specifically for the checkbox values.
REDCapR::checkbox_choices(select_choices=choices_1)

## Not run:
uri         <- "https://bbmc.ouhsc.edu/redcap/api/"
token       <- "9A81268476645C4E5F03428B8AC3AA7B"

ds_metadata <- redcap_metadata_read(redcap_uri=uri, token=token)$data
choices_2  <- ds_metadata[ds_metadata$field_name=="race", "select_choices_or_calculations"]

REDCapR::regex_named_captures(pattern=pattern_boxes, text=choices_2)

## End(Not run)

path_3      <- system.file(package="REDCapR", "test-data/project-simple/simple-metadata.csv")
ds_metadata_3  <- read.csv(path_3, stringsAsFactors=FALSE)
choices_3    <- ds_metadata_3[ds_metadata_3$field_name=="race", "select_choices_or_calculations"]
REDCapR::regex_named_captures(pattern=pattern_boxes, text=choices_3)
```

---

redcap_column_sanitize *Sanitize to adhere to REDCap character encoding requirements*

Description

Replace non-ASCII characters with legal characters that won't cause problems when writing to a REDCap project.

Usage

```
redcap_column_sanitize(d, column_names = colnames(d),
  encoding_initial = "latin1", substitution_character = "?")
```

Arguments

d	The base::data.frame() containing the dataset used to update the REDCap project. Required.
column_names	An array of character values indicating the names of the variables to sanitize. Optional.
encoding_initial	An array of character values indicating the names of the variables to sanitize. Optional.
substitution_character	The character value that replaces characters that were unable to be appropriately matched.

Details

Letters like an accented 'A' are replaced with a plain 'A'.

This is a thin wrapper around base::iconv(). The ASCII//TRANSLIT option does the actual transliteration work. As of R 3.1.0, the OSes use similar, but different, versions to convert the characters. Be aware of this in case you notice OS-dependent differences.

Value

A base::data.frame() with same columns, but whose character values have been sanitized.

Author(s)

Will Beasley

Examples

```
dirty <- data.frame(id=1:3, names=c("Ekstr\xf8m", "J\xf6reskog", "bi\xdfchen Z\xfcrcher"))
REDCapR::redcap_column_sanitize(dirty)
# Produces the dataset:
#  id            names
#1  1          Ekstr?m
#2  2         Joreskog
#3  3 bisschen Zurcher

# Typical examples are not shown because they require non-ASCII encoding,
#   which makes the package documentation less portable.
```

---

redcap_download_file_oneshot

Download a file from a REDCap project record

Description

This function uses REDCap's API to download a file.

Usage

```
redcap_download_file_oneshot(file_name = NULL, directory = NULL,
  overwrite = FALSE, redcap_uri, token, record, field, event = "",
  verbose = TRUE, config_options = NULL)
```

Arguments

file_name	The name of the file where the downloaded file is saved. If empty the original name of the file will be used and saved in the default directory. Optional.
directory	The directory where the file is saved. By default current directory. Optional
overwrite	Boolean value indicating if existing files should be overwritten. Optional
redcap_uri	The URI (uniform resource identifier) of the REDCap project. Required.
token	The user-specific string that serves as the password for a project. Required.
record	The record ID where the file is to be imported. Required
field	The name of the field where the file is saved in REDCap. Required
event	The name of the event where the file is saved in REDCap. Optional
verbose	A boolean value indicating if messages should be printed to the R console during the operation. Optional.
config_options	A list of options to pass to httr::POST() method in the 'httr' package. See the details below. Optional.

Details

Currently, the function doesn't modify any variable types to conform to REDCap's supported variables. See validate_for_write() for a helper function that checks for some common important conflicts.

Value

Currently, a list is returned with the following elements,

- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- records_affected_count: The number of records inserted or updated.
- affected_ids: The subject IDs of the inserted or updated records.
- elapsed_seconds: The duration of the function.
- raw_text: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the raw_text is returned as an empty string to save RAM.
- file_name: The name of the file persisted to disk. This is useful if the name stored in REDCap is used (which is the default).

Author(s)

Will Beasley, John J. Aponte

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri     <- "https://bbmc.ouhsc.edu/redcap/api/"
token   <- "D70F9ACD1EDD6F151C6EA78683944E98" #pid=213
record  <- 1
field   <- "mugshot"
# event <- "" # only for longitudinal events

result_1 <- REDCapR::redcap_download_file_oneshot(
  record        = record,
  field         = field,
  redcap_uri    = uri,
  token         = token
)
base::unlink("mugshot-1.jpg")

(full_name <- base::tempfile(pattern="mugshot", fileext=".jpg"))
result_2   <- REDCapR::redcap_download_file_oneshot(
  file_name     = full_name,
  record        = record,
  field         = field,
  redcap_uri    = uri,
  token         = token
)
base::unlink(full_name)

(relative_name <- "ssss.jpg")
result_3 <- REDCapR::redcap_download_file_oneshot(
  file_name    = relative_name,
  record       = record,
  field        = field,
  redcap_uri   = uri,
  token        = token
)
base::unlink(relative_name)

## End(Not run)
```

`redcap_metadata_read` *Export the metadata of a REDCap project*

Description

Export the metadata (as a data dictionary) of a REDCap project as a `base::data.frame()`. Each row in the data dictionary corresponds to one field in the project's dataset.

Usage

```
redcap_metadata_read(redcap_uri, token, forms = NULL,
  forms_collapsed = "", fields = NULL, fields_collapsed = "",
  verbose = TRUE, config_options = NULL)
```

Arguments

`redcap_uri`	The URI (uniform resource identifier) of the REDCap project. Required.
`token`	The user-specific string that serves as the password for a project. Required.
`forms`	An array, where each element corresponds to the REDCap form of the desired fields. Optional.
`forms_collapsed`	A single string, where the desired forms are separated by commas. Optional.
`fields`	An array, where each element corresponds to a desired project field. Optional.
`fields_collapsed`	A single string, where the desired field names are separated by commas. Optional.
`verbose`	A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
`config_options`	A list of options to pass to `POST` method in the `httr` package. See the details in `redcap_read_oneshot()`. Optional.

Details

Specifically, it internally uses multiple calls to `redcap_read_oneshot()` to select and return data. Initially, only primary key is queried through the REDCap API. The long list is then subsetted into partitions, whose sizes are determined by the `batch_size` parameter. REDCap is then queried for all variables of the subset's subjects. This is repeated for each subset, before returning a unified `base::data.frame()`.

The function allows a delay between calls, which allows the server to attend to other users' requests.

Value

Currently, a list is returned with the following elements:

- `data`: An R `base::data.frame()` of the desired records and columns.
- `success`: A boolean value indicating if the operation was apparently successful.
- `status_codes`: A collection of http status codes, separated by semicolons. There is one code for each batch attempted.
- `outcome_messages`: A collection of human readable strings indicating the operations' semicolons. There is one code for each batch attempted. In an unsuccessful operation, it should contain diagnostic information.
- `forms_collapsed`: The desired records IDs, collapsed into a single string, separated by commas.
- `fields_collapsed`: The desired field names, collapsed into a single string, separated by commas.
- `elapsed_seconds`: The duration of the function.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri   <- "https://bbmc.ouhsc.edu/redcap/api/"
token <- "9A81268476645C4E5F03428B8AC3AA7B"
REDCapR::redcap_metadata_read(redcap_uri=uri, token=token)

## End(Not run)
```

---

redcap_next_free_record_name

Determine free available record ID

Description

Determines the next available record ID.

Usage

```
redcap_next_free_record_name(redcap_uri, token, verbose = TRUE,
  config_options = NULL)
```

Arguments

redcap_uri	The URI (uniform resource identifier) of the REDCap project. Required.
token	The user-specific string that serves as the password for a project. Required.
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options to pass to POST method in the httr package. See the details below. Optional.

Details

If the API call is unsuccessful, a value of character(0) will be returned (*i.e.*, an empty vector). This ensures that a the function will always return an object of class base::character.

Value

a base::character vector of either length 1 (if successful) or length 0 (if not successful).

Note

Documentation in REDCap 8.4.0

To be used by projects with record auto-numbering enabled, this method exports the next potential record ID for a project. It generates the next record name by determining the current maximum numerical record ID and then incrementing it by one.

Note: This method does not create a new record, but merely determines what the next record name would be.

If using Data Access Groups (DAGs) in the project, this method accounts for the special formatting of the record name for users in DAGs (e.g., DAG-ID); in this case, it only assigns the next value for ID for all numbers inside a DAG. For example, if a DAG has a corresponding DAG number of 223 wherein records 223-1 and 223-2 already exist, then the next record will be 223-3 if the API user belongs to the DAG that has DAG number 223. (The DAG number is auto-assigned by REDCap for each DAG when the DAG is first created.) When generating a new record name in a DAG, the method considers all records in the entire project when determining the maximum record ID, including those that might have been originally created in that DAG but then later reassigned to another DAG.

Note: This method functions the same even for projects that do not have record auto-numbering enabled.

Examples

```
uri      <- "https://bbmc.ouhsc.edu/redcap/api/"
token    <- "9A81268476645C4E5F03428B8AC3AA7B"
REDCapR::redcap_next_free_record_name(redcap_uri=uri, token=token) # Should return "6"
```

`redcap_project` *A* `Reference Class` *to make later calls to REDCap more convenient*

Description

This `Reference Class` represents a REDCap project. Once some values are set that are specific to a REDCap project (such as the URI and token), later calls are less verbose (such as reading and writing data). The functionality

Fields

`redcap_uri` The URI (uniform resource identifier) of the REDCap project. Required.

`token` token The user-specific string that serves as the password for a project. Required.

Methods

`read(batch_size = 100L, interbatch_delay = 0, records = NULL, records_collapsed = "", fields = NULL, fiel`
Exports records from a REDCap project.

`write(ds_to_write, batch_size = 100L, interbatch_delay = 0, continue_on_error = FALSE, verbose = TRUE, co`
Imports records to a REDCap project.

Examples

```
uri     <- "https://bbmc.ouhsc.edu/redcap/api/"
token   <- "D70F9ACD1EDD6F151C6EA78683944E98"
## Not run:
project <- REDCapR::redcap_project$new(redcap_uri=uri, token=token)
ds_all  <- project$read()

#Demonstrate how repeated calls are more concise when the token and url aren't always passed.
ds_three_columns <- project$read(fields=c("record_id", "sex", "height"))$data

ids_of_males vv <- ds_three_columns$record_id[ds_three_columns$sex==1]
ids_of_shorties <- ds_three_columns$record_id[ds_three_columns$height < 40]

ds_males        <- project$read(records=ids_of_males, batch_size=2)$data
ds_shorties     <- project$read(records=ids_of_shorties)$data

#Switch the Genders
sex_original         <- ds_three_columns$sex
ds_three_columns$sex <- (1 - ds_three_columns$sex)
project$write(ds_three_columns)
```

```
#Switch the Genders back
ds_three_columns$sex <- sex_original
project$write(ds_three_columns)

## End(Not run)
```

`redcap_read` *Read records from a REDCap project in subsets, and stacks them together before returning a* `base::data.frame()`

Description

From an external perspective, this function is similar to `redcap_read_oneshot()`. The internals differ in that `redcap_read` retrieves subsets of the data, and then combines them before returning (among other objects) a single `base::data.frame()`. This function can be more appropriate than `redcap_read_oneshot()` when returning large datasets that could tie up the server.

Usage

```
redcap_read(batch_size = 100L, interbatch_delay = 0.5,
  continue_on_error = FALSE, redcap_uri, token, records = NULL,
  records_collapsed = "", fields = NULL, fields_collapsed = "",
  forms = NULL, forms_collapsed = "", events = NULL,
  events_collapsed = "", raw_or_label = "raw",
  raw_or_label_headers = "raw", export_checkbox_label = FALSE,
  export_survey_fields = FALSE, export_data_access_groups = FALSE,
  filter_logic = "", guess_type = TRUE, guess_max = 1000L,
  verbose = TRUE, config_options = NULL, id_position = 1L)
```

Arguments

`batch_size`	The maximum number of subject records a single batch should contain. The default is 100.
`interbatch_delay`	The number of seconds the function will wait before requesting a new subset from REDCap. The default is 0.5 seconds.
`continue_on_error`	If an error occurs while reading, should records in subsequent batches be attempted. The default is `FALSE`, which prevents subsequent batches from running. Required.
`redcap_uri`	The URI (uniform resource identifier) of the REDCap project. Required.
`token`	The user-specific string that serves as the password for a project. Required.
`records`	An array, where each element corresponds to the ID of a desired record. Optional.

`records_collapsed`
: A single string, where the desired ID values are separated by commas. Optional.

`fields`
: An array, where each element corresponds to a desired project field. Optional.

`fields_collapsed`
: A single string, where the desired field names are separated by commas. Optional.

`forms`
: An array, where each element corresponds to a desired project form. Optional.

`forms_collapsed`
: A single string, where the desired form names are separated by commas. Optional.

`events`
: An array, where each element corresponds to a desired project event. Optional.

`events_collapsed`
: A single string, where the desired event names are separated by commas. Optional.

`raw_or_label`
: A string (either 'rawor 'label' that specifies whether to export the raw coded values or the labels for the options of multiple choice fields. Default is'raw'‘.

`raw_or_label_headers`
: A string (either 'raw' or 'label' that specifies for the CSV headers whether to export the variable/field names (raw) or the field labels (label). Default is 'raw'.

`export_checkbox_label`
: specifies the format of checkbox field values specifically when exporting the data as labels. If `raw_or_label` is 'label' and `export_checkbox_label` is TRUE, the values will be the text displayed to the users. Otherwise, the values will be 0/1.

`export_survey_fields`
: A boolean that specifies whether to export the survey identifier field (e.g., 'redcap_survey_identifier') or survey timestamp fields (e.g., instrument+'_timestamp') .

`export_data_access_groups`
: A boolean value that specifies whether or not to export the `redcap_data_access_group` field when data access groups are utilized in the project. Default is `FALSE`. See the details below.

`filter_logic`
: String of logic text (e.g., `[gender] = 'male'`) for filtering the data to be returned by this API method, in which the API will only return the records (or record-events, if a longitudinal project) where the logic evaluates as TRUE. An blank/empty string returns all records.

`guess_type`
: A boolean value indicating if all columns should be returned as character. If true, `readr::read_csv()` guesses the intended data type for each column.

guess_max	A positive integer passed to readr::read_csv() **per batch** that specifies the maximum number of records to use for guessing column types.
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options to pass to POST method in the httr package. See the details in redcap_read_oneshot() Optional.
id_position	The column position of the variable that unique identifies the subject. This defaults to the first variable in the dataset.

Details

Specifically, it internally uses multiple calls to redcap_read_oneshot() to select and return data. Initially, only primary key is queried through the REDCap API. The long list is then subsetted into partitions, whose sizes are determined by the batch_size parameter. REDCap is then queried for all variables of the subset's subjects. This is repeated for each subset, before returning a unified base::data.frame().

The function allows a delay between calls, which allows the server to attend to other users' requests.

For redcap_read() to function properly, the user must have Export permissions for the 'Full Data Set'. Users with only 'De-Identified' export privileges can still use redcap_read_oneshot. To grant the appropriate permissions:

- go to 'User Rights' in the REDCap project site,
- select the desired user, and then select 'Edit User Privileges',
- in the 'Data Exports' radio buttons, select 'Full Data Set'.

Value

Currently, a list is returned with the following elements:

- data: An R base::data.frame() of the desired records and columns.
- success: A boolean value indicating if the operation was apparently successful.
- status_codes: A collection of http status codes, separated by semicolons. There is one code for each batch attempted.
- outcome_messages: A collection of human readable strings indicating the operations' semicolons. There is one code for each batch attempted. In an unsuccessful operation, it should contain diagnostic information.
- records_collapsed: The desired records IDs, collapsed into a single string, separated by commas.
- fields_collapsed: The desired field names, collapsed into a single string, separated by commas.
- filter_logic: The filter statement passed as an argument.
- elapsed_seconds: The duration of the function.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri     <- "https://bbmc.ouhsc.edu/redcap/api/"
token   <- "9A81268476645C4E5F03428B8AC3AA7B"
REDCapR::redcap_read(batch_size=2, redcap_uri=uri, token=token)

## End(Not run)
```

---

redcap_read_oneshot *Read/Export records from a REDCap project*

Description

This function uses REDCap's API to select and return data.

Usage

```
redcap_read_oneshot(redcap_uri, token, records = NULL,
  records_collapsed = "", fields = NULL, fields_collapsed = "",
  forms = NULL, forms_collapsed = "", events = NULL,
  events_collapsed = "", raw_or_label = "raw",
  raw_or_label_headers = "raw", export_checkbox_label = FALSE,
  export_survey_fields = FALSE, export_data_access_groups = FALSE,
  filter_logic = "", guess_type = TRUE, guess_max = 1000L,
  verbose = TRUE, config_options = NULL)
```

Arguments

redcap_uri	The URI (uniform resource identifier) of the REDCap project. Required.
token	The user-specific string that serves as the password for a project. Required.
records	An array, where each element corresponds to the ID of a desired record. Optional.
records_collapsed	A single string, where the desired ID values are separated by commas. Optional.

`fields` An array, where each element corresponds to a desired project field. Optional.

`fields_collapsed`
A single string, where the desired field names are separated by commas. Optional.

`forms` An array, where each element corresponds to a desired project form. Optional.

`forms_collapsed`
A single string, where the desired form names are separated by commas. Optional.

`events` An array, where each element corresponds to a desired project event. Optional.

`events_collapsed`
A single string, where the desired event names are separated by commas. Optional.

`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 multiple choice fields. Default is `'raw'`.

`raw_or_label_headers`
A string (either `'raw'` or `'label'` that specifies for the CSV headers whether to export the variable/field names (raw) or the field labels (label). Default is `'raw'`.

`export_checkbox_label`
specifies the format of checkbox field values specifically when exporting the data as labels. If `raw_or_label` is `'label'` and `export_checkbox_label` is TRUE, the values will be the text displayed to the users. Otherwise, the values will be 0/1.

`export_survey_fields`
A boolean that specifies whether to export the survey identifier field (e.g., 'redcap_survey_identifier') or survey timestamp fields (e.g., instrument+'_timestamp') .

`export_data_access_groups`
A boolean value that specifies whether or not to export the `redcap_data_access_group` field when data access groups are utilized in the project. Default is `FALSE`. See the details below.

`filter_logic` String of logic text (e.g., `[gender] = 'male'`) for filtering the data to be returned by this API method, in which the API will only return the records (or record-events, if a longitudinal project) where the logic evaluates as TRUE. An blank/empty string returns all records.

`guess_type` A boolean value indicating if all columns should be returned as character. If false, `readr::read_csv()` guesses the intended data type for each column.

`guess_max` A positive integer passed to `readr::read_csv()` that specifies the maximum number of records to use for guessing column types.

`verbose` A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive

information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.

`config_options` A list of options to pass to `POST` method in the `httr` package. See the details below. Optional.

Details

The full list of configuration options accepted by the `httr` package is viewable by executing `httr::httr_options()`. The `httr` package and documentation is available at https://cran.r-project.org/package=httr.

If you do not pass in this export_data_access_groups value, it will default to `FALSE`. The following is from the API help page for version 5.2.3: This flag is only viable if the user whose token is being used to make the API request is *not* in a data access group. If the user is in a group, then this flag will revert to its default value.

Value

Currently, a list is returned with the following elements:

- `data`: An R `base::data.frame()` of the desired records and columns.
- `success`: A boolean value indicating if the operation was apparently successful.
- `status_code`: The http status code of the operation.
- `outcome_message`: A human readable string indicating the operation's outcome.
- `records_collapsed`: The desired records IDs, collapsed into a single string, separated by commas.
- `fields_collapsed`: The desired field names, collapsed into a single string, separated by commas.
- `filter_logic`: The filter statement passed as an argument.
- `elapsed_seconds`: The duration of the function.
- `raw_text`: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the `raw_text` is returned as an empty string to save RAM.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri      <- "https://bbmc.ouhsc.edu/redcap/api/"
token    <- "9A81268476645C4E5F03428B8AC3AA7B"

#Return all records and all variables.
ds <- REDCapR::redcap_read_oneshot(redcap_uri=uri, token=token)$data

#Return only records with IDs of 1 and 3
desired_records_v1 <- c(1, 3)
ds_some_rows_v1 <- REDCapR::redcap_read_oneshot(
   redcap_uri = uri,
   token      = token,
   records    = desired_records_v1
)$data

#Return only the fields record_id, name_first, and age
desired_fields_v1 <- c("record_id", "name_first", "age")
ds_some_fields_v1 <- REDCapR::redcap_read_oneshot(
   redcap_uri = uri,
   token      = token,
   fields     = desired_fields_v1
)$data

## End(Not run)
```

---

redcap_read_oneshot_eav

Read/Export records from a REDCap project –still in development

---

Description

This function uses REDCap's API to select and return data. This function is still in development.

Usage

```
redcap_read_oneshot_eav(redcap_uri, token, records = NULL,
  records_collapsed = "", fields = NULL, fields_collapsed = "",
  forms = NULL, forms_collapsed = "", events = NULL,
  events_collapsed = "", raw_or_label = "raw",
  raw_or_label_headers = "raw", export_data_access_groups = FALSE,
  filter_logic = "", verbose = TRUE, config_options = NULL)
```

Arguments

redcap_uri	The URI (uniform resource identifier) of the REDCap project. Required.

`token` The user-specific string that serves as the password for a project. Required.

`records` An array, where each element corresponds to the ID of a desired record. Optional.

`records_collapsed`
A single string, where the desired ID values are separated by commas. Optional.

`fields` An array, where each element corresponds to a desired project field. Optional.

`fields_collapsed`
A single string, where the desired field names are separated by commas. Optional.

`forms` An array, where each element corresponds to a desired project field. Optional.

`forms_collapsed`
A single string, where the desired form names are separated by commas. Optional.

`events` An array, where each element corresponds to a desired project event. Optional.

`events_collapsed`
A single string, where the desired event names are separated by commas. Optional.

`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 multiple choice fields. Default is `'raw'`.

`raw_or_label_headers`
A string (either `'raw'` or `'label'` that specifies for the CSV headers whether to export the variable/field names (raw) or the field labels (label). Default is `'raw'`.

`export_data_access_groups`
A boolean value that specifies whether or not to export the `redcap_data_access_group` field when data access groups are utilized in the project. Default is `FALSE`. See the details below.

`filter_logic` String of logic text (e.g., `[gender] = 'male'`) for filtering the data to be returned by this API method, in which the API will only return the records (or record-events, if a longitudinal project) where the logic evaluates as TRUE. An blank/empty string returns all records.

`verbose` A boolean value indicating if `message`s should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.

`config_options` A list of options to pass to `POST` method in the `httr` package. See the details below. Optional.

Details

The full list of configuration options accepted by the `httr` package is viewable by executing `httr::httr_options()`. The `httr` package and documentation is available at https://cran.r-project.org/package=httr.

If you do not pass in this export_data_access_groups value, it will default to `FALSE`. The following is from the API help page for version 5.2.3: This flag is only viable if the user whose token is being used to make the API request is *not* in a data access group. If the user is in a group, then this flag will revert to its default value.

As of REDCap 6.14.3, this field is not exported in the EAV API call.

Value

Currently, a list is returned with the following elements:

- `data`: An R `base::data.frame()` of the desired records and columns.
- `success`: A boolean value indicating if the operation was apparently successful.
- `status_code`: The http status code of the operation.
- `outcome_message`: A human readable string indicating the operation's outcome.
- `records_collapsed`: The desired records IDs, collapsed into a single string, separated by commas.
- `fields_collapsed`: The desired field names, collapsed into a single string, separated by commas.
- `filter_logic`: The filter statement passed as an argument.
- `elapsed_seconds`: The duration of the function.
- `raw_text`: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the `raw_text` is returned as an empty string to save RAM.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri      <- "https://bbmc.ouhsc.edu/redcap/api/"
token    <- "9A81268476645C4E5F03428B8AC3AA7B"

#Return all records and all variables.
ds <- REDCapR:::redcap_read_oneshot_eav(redcap_uri=uri, token=token)$data
```

```
#Return only records with IDs of 1 and 3
desired_records_v1 <- c(1, 3)
ds_some_rows_v1    <- REDCapR::redcap_read_oneshot_eav(
   redcap_uri = uri,
   token      = token,
   records    = desired_records_v1
)$data

#Return only the fields record_id, name_first, and age
desired_fields_v1 <- c("record_id", "name_first", "age")
ds_some_fields_v1 <- redcap_read_oneshot_eav(
   redcap_uri = uri,
   token      = token,
   fields     = desired_fields_v1
)$data

## End(Not run)
```

`redcap_upload_file_oneshot`

Upload a file into to a REDCap project record

Description

This function uses REDCap's API to upload a file.

Usage

```
redcap_upload_file_oneshot(file_name, record, redcap_uri, token, field,
  event = "", verbose = TRUE, config_options = NULL)
```

Arguments

`file_name`	The name of the relative or full file to be uploaded into the REDCap project. Required.
`record`	The record ID where the file is to be imported. Required
`redcap_uri`	The URI (uniform resource identifier) of the REDCap project. Required.
`token`	The user-specific string that serves as the password for a project. Required.
`field`	The name of the field where the file is saved in REDCap. Required
`event`	The name of the event where the file is saved in REDCap. Optional
`verbose`	A boolean value indicating if `messages` should be printed to the R console during the operation. Optional.
`config_options`	A list of options to pass to `POST` method in the `httr` package. See the details below. Optional.

Details

Currently, the function doesn't modify any variable types to conform to REDCap's supported variables. See `validate_for_write()` for a helper function that checks for some common important conflicts.

Value

Currently, a list is returned with the following elements:

- `success`: A boolean value indicating if the operation was apparently successful.
- `status_code`: The http status code of the operation.
- `outcome_message`: A human readable string indicating the operation's outcome.
- `records_affected_count`: The number of records inserted or updated.
- `affected_ids`: The subject IDs of the inserted or updated records.
- `elapsed_seconds`: The duration of the function.
- `raw_text`: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the `raw_text` is returned as an empty string to save RAM.

Author(s)

Will Beasley, John J. Aponte

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (ie, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
#Define some constants
uri       <- "https://bbmc.ouhsc.edu/redcap/api/"
token     <- "D70F9ACD1EDD6F151C6EA78683944E98" #For the simple project (pid 213)
field     <- "mugshot"
event     <- "" # only for longitudinal events

#Upload a single image file.
record    <- 1
file_path <- system.file("test-data/mugshot-1.jpg", package="REDCapR")

REDCapR::redcap_upload_file_oneshot(
  file_name  = file_path, record=record, field=field,
  redcap_uri = redcap_uri, token=token
)

#Upload a collection of five images.
```

```
records    <- 1:5
file_paths <- system.file(paste0("test-data/mugshot-", records, ".jpg"), package="REDCapR")

for( i in seq_along(records) ) {
  record    <- records[i]
  file_path <- file_paths[i]
  REDCapR::redcap_upload_file_oneshot(
    file_name  = file_path, record=record, field=field,
    redcap_uri = redcap_uri, token=token
  )
}

## End(Not run)
```

`redcap_users_export` *List authorized users*

Description

List users authorized for a project.

Usage

```
redcap_users_export(redcap_uri, token, verbose = TRUE,
  config_options = NULL)
```

Arguments

`redcap_uri`	The URI (uniform resource identifier) of the REDCap project. Required.
`token`	The user-specific string that serves as the password for a project. Required.
`verbose`	A boolean value indicating if `messages` should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
`config_options`	A list of options to pass to `POST` method in the `httr` package. See the details below. Optional.

Value

a `utils::packageVersion`.

Note

Documentation in REDCap 8.4.0

This method allows you to export the list of users for a project, including their user privileges and also email address, first name, and last name.

Note: If the user has been assigned to a user role, it will return the user with the role's defined privileges.

Examples

```
uri      <- "https://bbmc.ouhsc.edu/redcap/api/"
token    <- "06DEFB601F9B46847DAA9DF0CFA951B4"
result   <- REDCapR::redcap_users_export(redcap_uri=uri, token=token)
result$data_user
result$data_user_form
```

---

redcap_variables *Enumerate the exported variables*

Description

This function calls the 'exportFieldNames' function of the REDCap API.

Usage

```
redcap_variables(redcap_uri, token, verbose = TRUE,
  config_options = NULL)
```

Arguments

redcap_uri	The URI (uniform resource identifier) of the REDCap project. Required.
token	The user-specific string that serves as the password for a project. Required.
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options to pass to POST method in the httr package. See the details below. Optional.

Details

The full list of configuration options accepted by the httr package is viewable by executing httr::httr_options(). The httr package and documentation is available at https://cran.r-project.org/package=httr.

As of REDCap version 6.14.2, three variable types are *not* returned in this call: calculated, file, and descriptive. All variables returned are writable/uploadable.

Value

Currently, a list is returned with the following elements,

- data: An R base::data.frame() where each row represents one column in the REDCap dataset.
- success: A boolean value indicating if the operation was apparently successful.

- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- elapsed_seconds: The duration of the function.
- raw_text: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the raw_text is returned as an empty string to save RAM.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
uri         <- "https://bbmc.ouhsc.edu/redcap/api/"
token       <- "9A81268476645C4E5F03428B8AC3AA7B"
ds_variable <- REDCapR::redcap_variables(redcap_uri=uri, token=token)$data

## End(Not run)
```

---

redcap_version *Determine version of REDCap instance*

Description

This function uses REDCap's API to query its version.

Usage

```
redcap_version(redcap_uri, token, verbose = TRUE,
  config_options = NULL)
```

Arguments

redcap_uri	The URI (uniform resource identifier) of the REDCap project. Required.
token	The user-specific string that serves as the password for a project. Required.
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.

config_options A list of options to pass to POST method in the httr package. See the details below. Optional.

Details

If the API call is unsuccessful, a value of base::package_version("0.0.0") will be returned. This ensures that a the function will always return an object of class base::package_version. It guarantees the value can always be used in utils::compareVersion().

Value

a utils::packageVersion.

Examples

```
uri      <- "https://bbmc.ouhsc.edu/redcap/api/"
token    <- "9A81268476645C4E5F03428B8AC3AA7B"
REDCapR::redcap_version(redcap_uri=uri, token=token)
```

---

redcap_write *Write/Import records to a REDCap project*

Description

This function uses REDCap's APIs to select and return data.

Usage

```
redcap_write(ds_to_write, batch_size = 100L, interbatch_delay = 0.5,
  continue_on_error = FALSE, redcap_uri, token, verbose = TRUE,
  config_options = NULL)
```

Arguments

ds_to_write The base::data.frame() to be imported into the REDCap project. Required.

batch_size The maximum number of subject records a single batch should contain. The default is 100.

interbatch_delay The number of seconds the function will wait before requesting a new subset from REDCap. The default is 0.5 seconds.

continue_on_error If an error occurs while writing, should records in subsequent batches be attempted. The default is FALSE, which prevents subsequent batches from running. Required.

redcap_uri The URI (uniform resource identifier) of the REDCap project. Required.

token The user-specific string that serves as the password for a project. Required.

verbose — A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.

config_options — A list of options to pass to POST method in the httr package. See the details in redcap_read_oneshot(). Optional.

Details

Currently, the function doesn't modify any variable types to conform to REDCap's supported variables. See validate_for_write() for a helper function that checks for some common important conflicts.

For redcap_write to function properly, the user must have Export permissions for the 'Full Data Set'. Users with only 'De-Identified' export privileges can still use redcap_write_oneshot(). To grant the appropriate permissions:

- go to 'User Rights' in the REDCap project site,
- select the desired user, and then select 'Edit User Privileges',
- in the 'Data Exports' radio buttons, select 'Full Data Set'.

Value

Currently, a list is returned with the following elements:

- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- records_affected_count: The number of records inserted or updated.
- affected_ids: The subject IDs of the inserted or updated records.
- elapsed_seconds: The duration of the function.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
#Define some constants
uri           <- "https://bbmc.ouhsc.edu/redcap/api/"
token         <- "D70F9ACD1EDD6F151C6EA78683944E98"

# Read the dataset for the first time.
result_read1  <- REDCapR::redcap_read_oneshot(redcap_uri=uri, token=token)
ds1           <- result_read1$data
ds1$telephone
# The line above returns something like this (depending on its previous state).
# [1] "(432) 456-4848" "(234) 234-2343" "(433) 435-9865" "(987) 654-3210" "(333) 333-4444"

# Manipulate a field in the dataset in a VALID way
ds1$telephone <- sprintf("(405) 321-%1$i%1$i%1$i%1$i", seq_len(nrow(ds1)))

ds1 <- ds1[1:3, ]
ds1$age       <- NULL; ds1$bmi <- NULL #Drop the calculated fields before writing.
result_write  <- REDCapR::redcap_write(ds=ds1, redcap_uri=uri, token=token)

# Read the dataset for the second time.
result_read2  <- REDCapR::redcap_read_oneshot(redcap_uri=uri, token=token)
ds2           <- result_read2$data
ds2$telephone
# The line above returns something like this.  Notice only the first three lines changed.
# [1] "(405) 321-1111" "(405) 321-2222" "(405) 321-3333" "(987) 654-3210" "(333) 333-4444"

# Manipulate a field in the dataset in an INVALID way.  A US exchange can't be '111'.
ds1$telephone <- sprintf("(405) 111-%1$i%1$i%1$i%1$i", seq_len(nrow(ds1)))

# This next line will throw an error.
result_write <- REDCapR::redcap_write(ds=ds1, redcap_uri=uri, token=token)
result_write$raw_text

## End(Not run)
```

---

redcap_write_oneshot *Write/Import records to a REDCap project*

Description

This function uses REDCap's API to select and return data.

Usage

```
redcap_write_oneshot(ds, redcap_uri, token, verbose = TRUE,
  config_options = NULL)
```

Arguments

ds	The base::data.frame() to be imported into the REDCap project. Required.
redcap_uri	The URI (uniform resource identifier) of the REDCap project. Required.
token	The user-specific string that serves as the password for a project. Required.
verbose	A boolean value indicating if messages should be printed to the R console during the operation. The verbose output might contain sensitive information (*e.g.* PHI), so turn this off if the output might be visible somewhere public. Optional.
config_options	A list of options to pass to httr::POST() method in the 'httr' package. See the details in redcap_read_oneshot() Optional.

Details

Currently, the function doesn't modify any variable types to conform to REDCap's supported variables. See validate_for_write() for a helper function that checks for some common important conflicts.

Value

Currently, a list is returned with the following elements:

- success: A boolean value indicating if the operation was apparently successful.
- status_code: The http status code of the operation.
- outcome_message: A human readable string indicating the operation's outcome.
- records_affected_count: The number of records inserted or updated.
- affected_ids: The subject IDs of the inserted or updated records.
- elapsed_seconds: The duration of the function.
- raw_text: If an operation is NOT successful, the text returned by REDCap. If an operation is successful, the raw_text is returned as an empty string to save RAM.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
## Not run:
#Define some constants
uri            <- "https://bbmc.ouhsc.edu/redcap/api/"
token          <- "D70F9ACD1EDD6F151C6EA78683944E98"

# Read the dataset for the first time.
result_read1   <- REDCapR::redcap_read_oneshot(redcap_uri=uri, token=token)
ds1            <- result_read1$data
ds1$telephone
# The line above returns something like this (depending on its previous state).
# [1] "(432) 456-4848" "(234) 234-2343" "(433) 435-9865" "(987) 654-3210" "(333) 333-4444"

# Manipulate a field in the dataset in a VALID way
ds1$telephone  <- sprintf("(405) 321-%1$i%1$i%1$i%1$i", seq_len(nrow(ds1)))

ds1 <- ds1[1:3, ]
ds1$age        <- NULL; ds1$bmi <- NULL #Drop the calculated fields before writing.
result_write   <- REDCapR::redcap_write_oneshot(ds=ds1, redcap_uri=uri, token=token)

# Read the dataset for the second time.
result_read2   <- REDCapR::redcap_read_oneshot(redcap_uri=uri, token=token)
ds2            <- result_read2$data
ds2$telephone
# The line above returns something like this.  Notice only the first three lines changed.
# [1] "(405) 321-1111" "(405) 321-2222" "(405) 321-3333" "(987) 654-3210" "(333) 333-4444"

# Manipulate a field in the dataset in an INVALID way.  A US exchange can't be '111'.
ds1$telephone  <- sprintf("(405) 111-%1$i%1$i%1$i%1$i", seq_len(nrow(ds1)))

# This next line will throw an error.
result_write   <- REDCapR::redcap_write_oneshot(ds=ds1, redcap_uri=uri, token=token)
result_write$raw_text

## End(Not run)
```

`replace_nas_with_explicit` *Create explicit factor level for missing values*

Description

Missing values are converted to a factor level. This explicit assignment can reduce the chances that missing values are inadvertently ignored. It also allows the presence of a missing to become a predictor in models.

Usage

```
replace_nas_with_explicit(scores, new_na_label = "Unknown",
  create_factor = FALSE, add_unknown_level = FALSE)
```

Arguments

`scores`	An array of values, ideally either factor or character. Required
`new_na_label`	The factor label assigned to the missing value. Defaults to `Unknown`.
`create_factor`	Converts `scores` into a factor, if it isn't one already. Defaults to `FALSE`.
`add_unknown_level`	Should a new factor level be created? (Specify `TRUE` if it already exists.) Defaults to `FALSE`.

Value

An array of values, where the `NA` values are now a factor level, with the label specified by the `new_na_label` value.

Note

The `create_factor` parameter is respected only if `scores` isn't already a factor. Otherwise, levels without any values would be lost.

A `stop` error will be thrown if the operation fails to convert all the `NA` values.

Author(s)

Will Beasley

Examples

```
library(REDCapR) #Load the package into the current R session.
```

`retrieve_credential` *Read a token and other credentials from a (non-REDCap) database or file*

Description

These functions are not essential to calling the REDCap API, but instead are functions that help manage tokens securely.

Usage

```
retrieve_credential_local(
  path_credential, project_id, check_url=TRUE,
  check_username=FALSE, check_token_pattern=TRUE
)
retrieve_credential_mssql(
  project_id, instance, dsn, channel=NULL
)
```

Arguments

path_credential	The file path to the CSV containing the credentials. Required.
project_id	The ID assigned to the project withing REDCap. This allows the user to store tokens to multiple REDCap projects in one file. Required
check_url	A logical value indicates if the url in the credential file should be checked to have approximately the correct form. Defaults to TRUE.
check_username	A logical value indicates if the username in the credential file should be checked against the username returned by R. Defaults to FALSE.
check_token_pattern	A logical value indicates if the token in the credential file is a 32-character hexadecimal string. Defaults to FALSE.
instance	The casual name associated with the REDCap instance on campus. This allows one credential system to accommodate multiple instances on campus. Required
dsn	A DSN on the local machine that points to the desired MSSQL database. Required.
channel	An *optional* connection handle as returned by DBI::dbConnect(). See Details below. Optional.

Details

If the database elements are created with the script provided in package's 'Security Database' vignette, the default values will work.

Value

A list of the following elements:

- redcap_uri: The URI of the REDCap Server.
- username: Username.
- project_id: The ID assigned to the project withing REDCap.
- token: The token to pass to the REDCap server
- comment: An optional string.

Note

Although we strongly encourage storing all the tokens on a central server (*e.g.*, see the retrieve_credential_mssql() function and the "SecurityDatabase" vignette), there are times when this approach is not feasible and the token must be stored locally. Please contact us if your institution is using something other than SQL Server, and would like help adapting this approach to your infrastructure.

Author(s)

Will Beasley

Examples

```
# ---- Local File Example ----------------------------
path <- system.file("misc/example.credentials", package="REDCapR")
(p1  <- REDCapR::retrieve_credential_local(path, 153L))
(p2  <- REDCapR::retrieve_credential_local(path, 212L))
```

---

sanitize_token *Validate and sanitize the user's REDCap token*

Description

Verifies the token is nonmissing and conforms to the legal pattern of a 32-character hexadecimal value. Trailing line endings are removed.

Usage

```
sanitize_token(token)
```

Arguments

token	The REDCap token. Required.

Value

The token, without a terminal newline character.

Note

Contact your institution's REDCap administrator for more information about your project-specific token.

Author(s)

Hao Zhu, Benjamin Nutter, Will Beasley

Examples

```
secret_token_1 <- "12345678901234567890123456ABCDEF"
secret_token_2 <- "12345678901234567890123456ABCDEF\n"
REDCapR::sanitize_token(secret_token_1)
REDCapR::sanitize_token(secret_token_2)
```

validate — *Inspect a* `base::data.frame()` *to anticipate problems before writing to a REDCap project*

Description

This set of functions inspect a `base::data.frame()` to anticipate problems before writing with REDCap's API.

Usage

```
validate_for_write( d )

validate_no_logical( data_types, stop_on_error )

validate_field_names( field_names, stop_on_error )
```

Arguments

`data_types`	The data types of the `base::data.frame()` corresponding to the REDCap project.
`stop_on_error`	If `TRUE`, an error is thrown for violations. Otherwise, a dataset summarizing the problems is returned.
`d`	The `base::data.frame()` containing the dataset used to update the REDCap project.
`field_names`	The names of the fields/variables in the REDCap project.

Details

All functions listed in the Usage section above inspect a specific aspect of the dataset. The `validate_for_write()` function executes all these individual validation checks. It allows the client to check everything with one call.

Value

A `tibble::tibble()`, where each potential violation is a row. The two columns are:

- `field_name`: The name of the `base::data.frame()` that might cause problems during the upload.
- `field_index`: The position of the field. (For example, a value of '1' indicates the first column, while a '3' indicates the third column.)
- `concern`: A description of the problem potentially caused by the `field`.
- `suggestion`: A *potential* solution to the concern.

Author(s)

Will Beasley

References

The official documentation can be found on the 'API Help Page' and 'API Examples' pages on the REDCap wiki (*i.e.*, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectredcap.org/articles/462/api-examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

Examples

```
d <- data.frame(
  record_id      = 1:4,
  flag_logical   = c(TRUE, TRUE, FALSE, TRUE),
  flag_Uppercase = c(4, 6, 8, 2)
)
REDCapR::validate_for_write(d = d)
```

Index