Package 'REDCapR'

March 19, 2017

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 0.9.7.9000

Date 2017-02-06

Author Will Beasley [aut, cre],
David Bard [ctb],
Thomas Wilson [ctb],
John J Aponte [ctb],
Rollie Parrish [ctb],
Benjamin Nutter [ctb],
Andrew Peters [ctb],
Hao Zhu [ctb]

Maintainer Will Beasley <wibeasley@hotmail.com>

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 data.table,
dplyr,
httr(>= 1.0.0.9000),
methods,
readr

Suggests devtools,
knitr,
rmarkdown,
RODBC,

RODBCext,
testthat (>= 0.9)

License GPL-2

LazyData TRUE

VignetteBuilder knitr

RoxygenNote 6.0.1

Roxygen list(markdown = TRUE)

R topics documented:

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

Description

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 ('OuhscBbmc/REDCapR')` (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.

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)
```

`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

```
library(REDCapR) #Load the package into the current R session.
create_batch_glossary(100, 50)
create_batch_glossary(100, 25)
create_batch_glossary(100, 3)
d <- data.frame(
  record_id = 1:100,
  iv        = sample(x=4, size=100, replace=TRUE),
  dv        = rnorm(n=100)
)
create_batch_glossary(nrow(d), batch_size=40)
```

---

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. 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

```
library(REDCapR) #Load the package into the current R session.
#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.
regex_named_captures(pattern=pattern_boxes, text=choices_1)

#This function is designed specifically for the checkbox values.
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"]

regex_named_captures(pattern=pattern_boxes, text=choices_2)

## End(Not run)
```

`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 slight 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 (ie, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectre examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

The official cURL site discusses the process of using SSL to verify the server being connected to.

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 <- 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 <- 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 <- 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 (ie, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectre 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:
library(REDCapR) #Load the package into the current R session.
uri   <- "https://bbmc.ouhsc.edu/redcap/api/"
token <- "9A81268476645C4E5F03428B8AC3AA7B"
redcap_metadata_read(redcap_uri=uri, token=token)

## End(Not run)
```

`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`
Exports records from a REDCap project.

`write(ds_to_write, batch_size = 100L, interbatch_delay = 0, verbose = TRUE, config_options = NULL)`
Imports records to a REDCap project.

Examples

```
library(REDCapR) #Load the package into the current R session.
uri <- "https://bbmc.ouhsc.edu/redcap/api/"
token <- "D70F9ACD1EDD6F151C6EA78683944E98"
## Not run:
project <- 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 = "",
  events = NULL, events_collapsed = "", export_data_access_groups = FALSE,
  filter_logic = "", raw_or_label = "raw", 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 a desired project field. Optional.

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

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

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

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.

raw_or_label A string (either 'rawor 'label' that specifies whether to export the raw coded values or th

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 (ie, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectre examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

The official cURL site discusses the process of using SSL to verify the server being connected to.

Examples

```
## Not run:
library(REDCapR) #Load the package into the current R session.
uri     <- "https://bbmc.ouhsc.edu/redcap/api/"
token   <- "9A81268476645C4E5F03428B8AC3AA7B"
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 = "",
  events = NULL, events_collapsed = "", export_data_access_groups = FALSE,
  filter_logic = "", raw_or_label = "raw", 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 a desired project field. Optional.
fields_collapsed	A single string, where the desired field names are separated by commas. Optional.
events	An array, where each element corresponds a desired project event Optional.
events_collapsed	A single string, where the desired event names are separated by commas. Optional.
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.
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'.
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 (ie, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectre examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

The official cURL site discusses the process of using SSL to verify the server being connected to.

Examples

```
## Not run:
library(REDCapR) #Load the package into the current R session.
uri      <- "https://bbmc.ouhsc.edu/redcap/api/"
token    <- "9A81268476645C4E5F03428B8AC3AA7B"
#Return all records and all variables.
ds_all_rows_all_fields <- 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 <- 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 <- redcap_read_oneshot(
   redcap_uri = uri,
   token      = token,
   fields     = desired_fields_v1
)$data


#Use the SSL cert file that come with the openssl package.
cert_location <- system.file("cacert.pem", package="openssl")
if( file.exists(cert_location) ) {
  config_options         <- list(cainfo=cert_location)
  ds_different_cert_file <- redcap_read_oneshot(
    redcap_uri     = uri,
    token          = token,
    config_options = config_options
  )$data
}

#Force the connection to use SSL=3 (which is not preferred, and possibly insecure).
config_options <- list(sslversion=3)
ds_ssl_3 <- redcap_read_oneshot(
  redcap_uri     = uri,
  token          = token,
  config_options = config_options
)$data

config_options <- list(ssl.verifypeer=FALSE)
ds_no_ssl <- redcap_read_oneshot(
   redcap_uri     = uri,
   token          = token,
   config_options = config_options
)$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.projectre examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

The official cURL site discusses the process of using SSL to verify the server being connected to.

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 <- base::file.path(devtools::inst(name="REDCapR"), paste0("test-data/mugshot-1.jpg"))

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 <- base::file.path(
  devtools::inst(name="REDCapR"),
  paste0("test-data/mugshot-", records, ".jpg")
)

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

## End(Not run)
```

`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 (ie, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectre examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

The official cURL site discusses the process of using SSL to verify the server being connected to.

Examples

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

# Read the dataset for the first time.
result_read1  <- 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  <- 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 (ie, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectre examples.html). If you do not have an account for the wiki, please ask your campus REDCap administrator to send you the static material.

The official cURL site discusses the process of using SSL to verify the server being connected to.

Examples

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

# Read the dataset for the first time.
result_read1   <- 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   <- 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 `RODBC::odbcConnect()`. 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

```
library(REDCapR) #Load the package into the current R session.
# ---- Local File Example ----------------------------
path <- system.file("misc/example.credentials", package="REDCapR")
(p1  <- retrieve_credential_local(path, 153L))
(p2  <- retrieve_credential_local(path, 212L))
```

`retrieve_token` *Read a token from a (non-REDCap) database.*

Description

This function will soon be deprecated; please transition to `retrieve_token_mssql()`. These functions are not essential to calling the REDCap API, but instead are functions that help manage tokens securely.

Usage

```
retrieve_token_mssql(project_name, dsn = NULL, channel = NULL)
```

Arguments

`project_name`	The friendly/shortened name given to the REDCap project in the MSSQL table. Notice this isn't necessarily the same name used by REDCap. Required
`dsn`	A DSN on the local machine that points to the desired MSSQL database. Required.
`channel`	An *optional* connection handle as returned by `RODBC::odbcConnect()`. See Details below. Optional.

Details

If no `channel` is passed, one will be created at the beginning of the function, and destroyed at the end. However if a channel is created, it's the caller's responsibility to destroy this resource. If you're making successive calls to the database, it might be quicker to create a single `channel` object and batch the calls together. Otherwise, the performance should be equivalent.

If you create the `channel` object yourself, consider wrapping calls in a `base::tryCatch` block, and closing the channel in its `finally` expression; this helps ensure the expensive database resource isn't held open unnecessarily. See the internals of `retrieve_token_mssql()` for an example of closing the `channel` in a `tryCatch` block.

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

Value

The token, which is a 32 character string.

Note

We use Microsoft SQL Server, because that fits our University's infrastructure the easiest. But this approach theoretically can work with any LDAP-enabled database server. Please contact us if your institution is using something other than SQL Server, and would like help adapting this approach to your infrastructure.

There's a lot of error checking for SQL injection, but remember that the user is executing under their own credentials, so this doesn't obviate the need for disciplined credential management. There's nothing that can be done with this R function that isn't already exposed by any other interface into the database (eg, SQL Server Management Studio, or MySQL Workbench.)

Author(s)

Will Beasley

Examples

```
library(REDCapR) #Load the package into the current R session.
## Not run:
# ---- SQL Server Example ----------------------------
# Rely on `retrieve_token()` to create & destory the channel.
dsn      <- "TokenSecurity"
project  <- "DiabetesSurveyProject"
token    <- retrieve_token(dsn=dsn, project_name=project)

# Create & close the channel yourself, to optimize repeated calls.
dsn      <- "TokenSecurity"
project1 <- "DiabetesSurveyProject1"
project2 <- "DiabetesSurveyProject2"
project3 <- "DiabetesSurveyProject3"

channel  <- RODBC::odbcConnect(dsn=dsn)
token1   <- retrieve_token(dsn=dsn, project_name=project1)
```

```
token2   <- retrieve_token(dsn=dsn, project_name=project2)
token3   <- retrieve_token(dsn=dsn, project_name=project3)
RODBC::odbcClose(channel)

## End(Not run)
```

`validate_for_write` *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( d )

validate_no_uppercase( d )
```

Arguments

`d`	The `base::data.frame()` containing the dataset used to update the REDCap project. Required.

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 `base::data.frame()`, 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 (ie, https://community.projectredcap.org/articles/456/api-documentation.html and https://community.projectre 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)
)
validate_for_write(d = d)
```

Index