Package 'redcapAPI'

January 26, 2023

Type Package

Title Interface to 'REDCap'

Version 2.3.3

Maintainer Shawn Garbett <shawn.garbett@vumc.org>

Description Access data stored in 'REDCap' databases using the Application Programming Interface (API). 'REDCap' (Research Electronic Data CAPture; <https://projectredcap.org>) is a web application for building and managing online surveys and databases developed at Vanderbilt University. The API allows users to access data and project meta data (such as the data dictionary) from the web programmatically. The 'redcapAPI' package facilitates the process of accessing data with options to prepare an analysis-ready data set consistent with the definitions in a database's data dictionary.

License GPL-2

Depends R (>= 3.0.0)

Imports checkmate, chron, httr, labelVector, lubridate, stringr, tidyr

LazyLoad yes

URL https://github.com/vubiostat/redcapAPI,https://projectredcap.org

BugReports https://github.com/vubiostat/redcapAPI/issues

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Shawn Garbett [cre, ctb],
Benjamin Nutter [ctb, aut],
Stephen Lane [ctb],
Will Beasley [ctb],
Jeffrey Horner [aut],
Will Gray [ctb],
Jeremy Stephens [ctb],
Marcus Lehr [ctb]

Repository CRAN

Date/Publication 2023-01-26 09:40:02 UTC

R topics documented:

`allocationTable` *Allocation Tables for the Randomization Module*

Description

Generate allocation table for the REDCap randomization module. Randomization may be stratified by other (categorical) variables in the data set as well as by data access group. Additionally, randomization may be blocked to ensure balanced groups throughout the allocation

Usage

```
allocationTable(
  rcon,
  random,
  strata = NULL,
  group = NULL,
  dag.id = NULL,
  replicates,
  block.size,
  block.size.shift = 0,
  seed.dev = NULL,
  seed.prod = NULL,
  bundle = NULL,
  weights = NULL,
  ...
)

## S3 method for class 'redcapDbConnection'
allocationTable(
  rcon,
  random,
  strata = NULL,
  group = NULL,
  dag.id = NULL,
  replicates,
  block.size,
  block.size.shift = 0,
  seed.dev = NULL,
  seed.prod = NULL,
  bundle = NULL,
  weights = c(1, 1),
  ...
)

## S3 method for class 'redcapApiConnection'
allocationTable(
  rcon,
```

```
  random,
  strata = NULL,
  group = NULL,
  dag.id = NULL,
  replicates,
  block.size,
  block.size.shift = 0,
  seed.dev = NULL,
  seed.prod = NULL,
  bundle = NULL,
  weights = c(1, 1),
  ...
)

makeChoices(random_levels, block.size, weights)

allocationTable_offline(
  meta_data,
  random,
  strata = NULL,
  group = NULL,
  dag.id = NULL,
  replicates,
  block.size,
  block.size.shift = 0,
  seed.dev = NULL,
  seed.prod = NULL,
  bundle = NULL,
  weights = c(1, 1),
  ...
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection
random	The field name to be randomized.
strata	Field names by which to stratify the randomization.
group	A field name giving a group by which randomization should be stratified. This could also be listed in strata, but the argument is provided to remain consistent with the REDCap user interface.
dag.id	Data Access Group IDs. See the package wiki for instructions on how to get the ID's. (They cannot currently be accessed via the API)
replicates	The number of randomizations to perform within each stratum
block.size	Block size for the randomization. Blocking is recommended to ensure balanced groups throughout the randomization. This may be a vector to indicate variable block sizes throughout the randomization.

`block.size.shift`
: A vector the same length as `block.size` where the first element is 0. This controls when the block size changes as a proportion of the total sample size. When `block.size=c(8, 4, 2)` and `block.size.shift = c(0, .5, .9)`, the first half of the randomization is performed in blocks of 8, then the next 40 percent of the randomization is performed in blocks of 4, with the last 10 percent performed in blocks of 2.

`seed.dev`
: At least one value is required. If only one value is given, it will be converted to a vector with length equal to the number of strata. Values will be incremented by 100 to provide independent randomizations. This may also have length equal to the number of strata.

`seed.prod`
: Same as `seed.prod`, but used to seed the production allocation. No pairwise elements of `seed.dev` and `seed.prod` may be equal. This guarantees that the two randomization schemes are unique.

`bundle`
: A `redcapBundle` object.

`weights`
: An optional vector giving the sampling weights for each of the randomization groups. There must be one number for each level of the randomization variable. If named, the names must match the group labels. If unnamed, the group labels will be assigned in the same order they appear in the data dictionary. The weights will be normalized, so they do not need to sum to 1.0. In other words, `weights=c(3, 1)` can indicate a 3:1 sampling ratio.

`...`
: Arguments to be passed to other methods

`random_levels`
: A vector of the randomization group level names. Determined from the data dictionary.

`meta_data`
: A text string giving the location of the data dictionary downloaded from REDCap.

Details

Each element in `block.size` must be a multiple of the number of groups in the randomized variable.

The 'offline' version of the function operates on the data dictionary file downloaded from REDCap. This is made available for instances where the API can not be accessed for some reason (such as waiting for API approval from the REDCap administrator).

The value of `replicates` controls how many allocations are generated. It is possible to get slightly more replicates than requested if your blocking design cannot exactly match replicates. For example, if you as for 30 replicates in blocks of 8, a warning will be printed and you will receive 32 replicates in the randomization table.

Author(s)

Benjamin Nutter

References

More instruction on using `redcapAPI` to produce allocation tables is on the package wiki: https://github.com/nutterb/redcapAPI/wiki/Randomization-Module

Please refer to your institution's API documentation.

Additional details on API parameters are found on the package wiki at https://github.com/nutterb/redcapAPI/wiki/REDCap-API-Parameters

---

`apiCall` *Execute a Call to the REDCap API*

Description

`apiCall` is a wrapper for attempting to access the API via `httr::POST`, and then via `RCurl::postForm` if a particular error occurs. This prevents a particular kind of error for which I haven't found a proper solution, but at least allows the expected behavior of the package. Since writing this, I have found a solution to set the config option `encoding='identity'`. I may remove this function at some point in the future.

Usage

```
apiCall(url, body, config)
```

Arguments

`url`	URL of the REDCap API
`body`	List of parameters to be passed to `httr::POST`'s `body` argument or `RCurl::postForm`'s `.param` argument.
`config`	A list of options to be passed to `httr::POST`'s `config` argument or `RCurl::postForm`'s `.opts` argument.

Details

Somewhere in the middle of an upgrade to RStudio, R 3.1.1, and various other system changes, I began seeing the error 'GnuTLS recv error (-9): A TLS packet with unexpected length was received.' I still don't know what this error means, but it only occurs when using `httr` on Linux. The `RCurl` equivalents appear to work just fine.

In order to prevent this error from occurring, and making the package rather useless, `apiCall` wraps `httr::POST` into a `tryCatch` call. If the GnuTLS error is thrown, `apiCall` then resorts to using the `RCurl` equivalent call.

Since originally writing this function, I've determined that the problem occurs due to weird characters being exported from REDCap that cannot be properly escaped in R. It can be resolved by using the config option `encoding = 'identity'`. Making this a default could make this function unnecessary.

Author(s)

Benjamin Nutter

References

Please refer to your institution's API documentation.

Additional details on API parameters are found on the package wiki at https://github.com/nutterb/redcapAPI/wiki/REDCap-API-Parameters

---

`checkbox_suffixes` *Checkbox Suffixes*

Description

Checkbox variables return one vector of data for each option defined in the variable. The variables are returned with the suffix `___[option]`. `exportRecords` needs these suffixes in order to retrieve all of the variables and to apply the correct labels.

Usage

```
checkbox_suffixes(fields, meta_data, version)
```

Arguments

`fields`	The current field names of interest
`meta_data`	The meta data data frame.
`version`	The REDCap version number.

`cleanseMetaData` *Clean Meta Data of UTF Characters*

Description

There have been isolated cases observed where certain characters in the data dictionary prevent it from being downloaded correctly. In one case, the data dictionary could not be downloaded at all through the API. It is suspected that these problematic characters are a result of copying and pasting text out of word processing programs. The problematic characters are not necessarily visible and their exact location can be difficult to identify. As a last resort, `cleanseMetaData` can read a meta data file downloaded through the user interface, purge it of any UTF-8 characters, and write an alternate data dictionary that contains only ASCII characters.

Usage

```
cleanseMetaData(meta_data_file, meta_data_clean, overwrite = FALSE)
```

Arguments

meta_data_file	character(1) the path to a meta data file that has been downloaded using the REDCap user interface.
meta_data_clean	character(1) the path of the file to which the cleaned meta data will be written.
overwrite	logical(1) Permit the new file to overwrite an existing file

---

deleteArms *Delete Arms From a Project*

Description

Delete arms from a project. This is a destructive action that will result in deletion of any events and data associated with the arm. Due to its destructive nature, it may only be performed on databases in development status. Data loss is non-reversible.

Usage

```
deleteArms(rcon, arms, ...)

## S3 method for class 'redcapDbConnection'
deleteArms(rcon, arms, ...)

## S3 method for class 'redcapApiConnection'
deleteArms(
  rcon,
  arms,
  ...,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection.
arms	integerish, a vector of arm numbers that will be deleted.
...	Additional arguments to pass to other methods.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Value

None.

REDCap API Documentation

This method allows you to delete Arms from a project. Notice: Because of this method's destructive nature, it is only available for use for projects in Development status. Additionally, please be aware that deleting an arm also automatically deletes all events that belong to that arm, and will also automatically delete any records/data that have been collected under that arm (this is non-reversible data loss).

NOTE: This only works for longitudinal projects.

REDCap Version

At least 8.1.17+ (and likely some earlier versions)

References

Please refer to your institution's API documentation.

---

deleteFiles *Delete a File attached to a Record*

Description

This function allows you to remove a document that has been attached to an individual record

Usage

```
deleteFiles(rcon, record, field, event, ...)

## S3 method for class 'redcapDbConnection'
deleteFiles(rcon, record, field, event, ...)

## S3 method for class 'redcapApiConnection'
deleteFiles(
  rcon,
  record = NULL,
  field = NULL,
  event = NULL,
  ...,
  bundle = getOption("redcap_bundle"),
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection
record	The record ID in which the desired file is stored. Must be length 1.
field	The field name in which the file is stored. Must be length 1.

event	The event name for the file. Must be length 1. This applies only to longitudinal projects. If the event is not supplied for a longitudinal project, the API will return an error message.
...	Arguments to be passed to other methods
bundle	A redcapBundle object as created by exportBundle.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Author(s)

Benjamin Nutter

References

Please refer to your institution's API documentation.

Additional details on API parameters are found on the package wiki at https://github.com/nutterb/redcapAPI/wiki/REDCap-API-Parameters

---

deleteRecords *Delete Records From a Project*

Description

Delete records from a project. This is a destructive action that will result in deletion of any events and data associated with the arm. Data loss is non-reversible. The user must have 'Delete Record' privileges in the database.

Usage

```
deleteRecords(rcon, records, arms = NULL, ...)

## S3 method for class 'redcapDbConnection'
deleteRecords(rcon, records, arms = NULL, ...)

## S3 method for class 'redcapApiConnection'
deleteRecords(
  rcon,
  records,
  arms = NULL,
  ...,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection.
records	a vector of record names to be deleted.
arms	integerish, the arm number of the arm in which the record(s) should be deleted. (This can only be used if the project is longitudinal with more than one arm.) NOTE: If the arm parameter is not provided, the specified records will be deleted from all arms in which they exist. Whereas, if arm is provided, they will only be deleted from the specified arm.
...	Additional arguments to pass to other methods.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Value

The number of deleted records.

REDCap API Documentation

This method allows you to delete one or more records from a project in a single API request.

REDCap Version

At least 8.1.17+ (and likely some earlier versions)

References

Please refer to your institution's API documentation.

---

deprecated_redcapProjectInfo

Deprecated Functions

Description

The redcapProjectInfo function has been deprecated to avoid confusion with the API method now executed by exportProjectInformation. The replacement function is exportBundle.

Usage

```
redcapProjectInfo(
  rcon,
  date = TRUE,
  label = TRUE,
  meta_data = TRUE,
  users = TRUE,
  instruments = TRUE,
  events = TRUE,
```

```
  arms = TRUE,
  mappings = TRUE,
  version = TRUE,
  ...
)

## S3 method for class 'redcapDbConnection'
redcapProjectInfo(
  rcon,
  date = TRUE,
  label = TRUE,
  meta_data = TRUE,
  users = TRUE,
  instruments = TRUE,
  events = TRUE,
  arms = TRUE,
  mappings = TRUE,
  version = TRUE,
  ...
)

## S3 method for class 'redcapApiConnection'
redcapProjectInfo(
  rcon,
  date = TRUE,
  label = TRUE,
  meta_data = TRUE,
  users = TRUE,
  instruments = TRUE,
  events = TRUE,
  arms = TRUE,
  mappings = TRUE,
  version = TRUE,
  ...,
  v.number = ""
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection
date	Logical. If TRUE, user expiration dates are converted to POSIXct objects.
label	Logical. If TRUE, the user form permissions are converted to labelled factors.
meta_data	Logical. Indicates if the meta data (data dictionary) should be exported.
users	Logical. Indicates if the users table should be exported.
instruments	Logical. Indicates if the instruments table should be exported.
events	Logical. Indicates if the event names should be exported.
arms	Logical. Indicates if the arms table should be exported.

mappings	Logical. Indicates if the form-event mappings should be exported.
version	Indicates if the REDCap version number should be exported. Only applicable in REDCap 6.0.0 and higher.
...	Arguments to be passed to other methods
v.number	A character string given the desired version number should the API method not be available.

---

exportArms *Export the Arms for a Project*

Description

This function allows you to export the Arms for a project Note: this only works for longitudinal projects

Usage

```
exportArms(rcon, ...)

## S3 method for class 'redcapDbConnection'
exportArms(rcon, ...)

## S3 method for class 'redcapApiConnection'
exportArms(
  rcon,
  arms = NULL,
  ...,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection.
...	Arguments to be passed to other methods.
arms	A numeric vector or arm numbers to retrieve. In REDCap 6.5.0, using this argument results in an empty data frame being returned.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Details

It is not sufficient to make the project a longitudinal project. The project must satisfy one of two conditions: 1) have at least two arms and one event defined in each arm; or 2) have one arm and at least two events defined. If neither of these conditions are satisfied, the API will return a message such as ERROR: You cannot export arms for classic projects, an error message that isn't as descriptive of the nature of the problem as we might like.

Value

Returns a data frame with two columns

- `arm_num` The arm number
- `name` The arm's descriptive name

REDCap API Documentation

This function allows you to export the Arms for a project

NOTE: this only works for longitudinal projects.

REDCap Version

5.8.2+

Known REDCap Limitations

In versions earlier than 5.9.15, providing a value to the `arms` argument had no effect and the entire data frame of arms is returned.

This was fixed in version 5.9.15. Sometime before 6.5.0, using the `arms` argument resulted in empty data frames being returned.

In most cases, the number or arms is fairly small, so there is no real performance benefit to only selecting a subset of the arms. The safest course of action is to export all of the arms (the default behavior)

Author(s)

Benjamin Nutter

References

Please refer to your institution's API documentation.

Additional details on API parameters are found on the package wiki at https://github.com/nutterb/redcapAPI/wiki/REDCap-API-Parameters

---

`exportBundle` *Perform a bundle of API calls.*

Description

Several of the API calls return objects that can be used to perform various validations in `exportRecords`, `exportReports`, and other methods. Using an export bundle allows you to call these methods once and store the result instead of issuing an additional call to the API each time a method is invoked.

For example, if you are uploading several files to the API, without an export bundle, `importFiles` will utilize the `exportMetaData` on each call in order to perform validations. Using a bundle allows you to download the meta data once and refer to it on every subsequent call that requires the data dictionary.

Usage

```
exportBundle(
  rcon,
  date = TRUE,
  label = TRUE,
  meta_data = TRUE,
  users = TRUE,
  instruments = TRUE,
  events = TRUE,
  arms = TRUE,
  mappings = TRUE,
  version = TRUE,
  ...
)

## S3 method for class 'redcapDbConnection'
exportBundle(
  rcon,
  date = TRUE,
  label = TRUE,
  meta_data = TRUE,
  users = TRUE,
  instruments = TRUE,
  events = TRUE,
  arms = TRUE,
  mappings = TRUE,
  version = TRUE,
  ...
)

## S3 method for class 'redcapApiConnection'
exportBundle(
  rcon,
  date = TRUE,
  label = TRUE,
  meta_data = TRUE,
  users = TRUE,
  instruments = TRUE,
  events = TRUE,
  arms = TRUE,
  mappings = TRUE,
  version = TRUE,
  ...,
  return_object = TRUE
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection
date	Logical. If TRUE, user expiration dates are converted to POSIXct objects.
label	Logical. If TRUE, the user form permissions are converted to labelled factors.
meta_data	Logical. Indicates if the meta data (data dictionary) should be exported.
users	Logical. Indicates if the users table should be exported.
instruments	Logical. Indicates if the instruments table should be exported.
events	Logical. Indicates if the event names should be exported.
arms	Logical. Indicates if the arms table should be exported.
mappings	Logical. Indicates if the form-event mappings should be exported.
version	Indicates if the REDCap version number should be exported. Only applicable in REDCap 6.0.0 and higher.
...	Arguments to be passed to other methods
return_object	Logical. When TRUE, the exportBundle object is returned to the workspace.

Details

The project information is stored in the option redcap_project_info. If the project is not longitudinal, the events, arms, and event-form mappings elements will be assigned character vectors instead of data frames.

Author(s)

Benjamin Nutter

---

exportEvents *Export the Events for a Project*

Description

Retrieve a data frame giving the users, expiration dates, and data access privileges for each user.

Usage

```
exportEvents(rcon, ...)

## S3 method for class 'redcapDbConnection'
exportEvents(rcon, arms = NULL, ...)

## S3 method for class 'redcapApiConnection'
exportEvents(
  rcon,
  arms = NULL,
  ...,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection.
...	Arguments to be passed to other methods.
arms	A numeric vector or arm numbers to retrieve.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Details

It is not sufficient to make the project a longitudinal project. The project must satisfy one of two conditions: 1) have at least two arms and one event defined; or 2) have one arm and at least two events defined. If neither of these conditions are satisfied, the API will return a message such as ERROR: You cannot export arms for classic projects, an error message that isn't as descriptive of the nature of the problem as we might like.

Value

Returns a data frame with six columns

- event_name The desciptive name of the event.
- arm_num The arm number in which the event occurs.
- day_offset The days offset from the first event.
- offset_min The minimum offset value.
- offset_max The maximium offset value.
- unique_event_name A unique event identifying name.

REDCap API Documentation

This function allows you to export the events for a project

NOTE: this only works for longitudinal projects.

REDCap Version

5.8.2+

Known REDCap Limitations

None

Author(s)

Benjamin Nutter

References

Please refer to your institution's API documentation.

Additional details on API parameters are found on the package wiki at https://github.com/nutterb/redcapAPI/wiki/REDCap-API-Parameters

`exportFieldNames` *Export the Export Field Names for a Project*

Description

Retrieve a data frame giving the original (as defined in REDCap) field name, choice values (for checkboxes), and the export field name.

Usage

```
exportFieldNames(
  rcon,
  ...,
  error_handling = getOption("redcap_error_handling")
)

## S3 method for class 'redcapDbConnection'
exportFieldNames(
  rcon,
  fields = NULL,
  bundle = NULL,
  ...,
  error_handling = getOption("redcap_error_handling")
)

## S3 method for class 'redcapApiConnection'
exportFieldNames(
  rcon,
  fields = NULL,
  bundle = NULL,
  ...,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

`rcon`	A REDCap connection object as generated by `redcapConnection`.
`...`	Arguments to be passed to other methods.
`error_handling`	An option for how to handle errors returned by the API. see `redcap_error`
`fields`	Field name to be returned. If `NULL`, all fields are returned.
`bundle`	A `redcapProject` object as created by `redcapProjectInfo`.

Value

A data frame containing three fields:

- `original_field_name` The field name as recorded in the data dictionary
- `choice_value` represents the raw coded value for a checkbox choice. For non-checkbox fields, this will always be `NA`.
- `export_field_name` The field name specific to the field. For non-checkbox fields, this is the same as `original_field_name`. For checkbox fields, it is the field name appended with `___[choice_value]`.

REDCap API Documentation

This function returns a list of the export/import-specific version of field names for all fields (or for one field, if desired) in a project. This is mostly used for checkbox fields because during data exports and data imports, checkbox fields have a different variable name used than the exact one defined for them in the Online Designer and Data Dictionary, in which *each checkbox option* gets represented as its own export field name in the following format: field_name + triple underscore + converted coded value for the choice. For non-checkbox fields, the export field name will be exactly the same as the original field name. Note: The following field types will be automatically removed from the list returned by this method since they cannot be utilized during the data import process: "calc", "file", and "descriptive".

The list that is returned will contain the three following attributes for each field/choice: "original_field_name", "choice_value", and "export_field_name". The choice_value attribute represents the raw coded value for a checkbox choice. For non-checkbox fields, the choice_value attribute will always be blank/empty. The export_field_name attribute represents the export/import-specific version of that field name.

REDCap Version

6.5.0+ (perhaps earlier; need to confirm its introduction)

Known REDCap Limitations

In 6.5.0, it has been observed that "slider" fields are not returned.

Signature fields are also not included, but these are effectively the same as "file" fields. This isn't a true limitation, but is documented here just to avoid confusion.

Author(s)

Stephen Lane

References

Please refer to your institution's API documentation (https://YOUR_REDCAP_URL/redcap/api/help)

Additional details on API parameters are found on the package wiki at https://github.com/nutterb/redcapAPI/wiki/REDCap-API-Parameters

exportFiles *Exports a File attached to a Record*

Description

A single file from a single record is retrieved. The behavior of this function is consistent with the behavior of the API, which only allows one file to be downloaded at a time

Usage

```
exportFiles(
  rcon,
  record,
  field,
  event,
  dir,
  filePrefix = TRUE,
  ...,
  bundle = getOption("redcap_bundle")
)

## S3 method for class 'redcapDbConnection'
exportFiles(
  rcon,
  record,
  field,
  event,
  dir,
  filePrefix = TRUE,
  ...,
  bundle = getOption("redcap_bundle")
)

## S3 method for class 'redcapApiConnection'
exportFiles(
  rcon,
  record,
  field,
  event = NULL,
  dir,
  filePrefix = TRUE,
  ...,
  bundle = getOption("redcap_bundle"),
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection
record	The record ID in which the desired file is stored. Must be length 1.
field	The field name in which the file is stored. Must be length 1.
event	The event name for the file. Must be length 1. This applies only to longitudinal projects. If the event is not supplied for a longitudinal project, the API will return an error message
dir	A directory/folder to which the file will be saved. By default, the working directory is used
filePrefix	Logical. Determines if a prefix is appended to the file name. The prefix takes the form [record_id]-[event_name]-[file_name]. The file name is always the same name of the file as it exists in REDCap
...	Arguments to be passed to other methods
bundle	A redcapBundle object as created by exportBundle.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Details

The function may only export a single file. See the examples for suggestions on exporting multiple files.

Note that the name of the file can not be changed. Whatever name exists in REDCap is the name that will be used, although the record ID and event name may be appended as a prefix

REDCap API Documentation (6.5.0)

This method allows you to download a document that has been attached to an individual record for a File Upload field. Please note that this method may also be used for Signature fields (i.e. File Upload fields with "signature" validation type).

Note about export rights: Please be aware that Data Export user rights will be applied to this API request. For example, if you have "No Access" data export rights in the project, then the API file export will fail and return an error. And if you have "De-Identified" or "Remove all tagged Identifier fields" data export rights, then the API file export will fail and return an error *only if* the File Upload field has been tagged as an Identifier field. To make sure that your API request does not return an error, you should have "Full Data Set" export rights in the project.

REDCap Version

5.8.2+

Known REDCap Limitations

None

Author(s)

Benjamin Nutter

References

Please refer to your institution's API documentation.

Additional details on API parameters are found on the package wiki at https://github.com/nutterb/redcapAPI/wiki/REDCap-API-Parameters

---

`exportInstruments` *Exports the REDCap Instruments*

Description

Returns a data frame of instruments, names, etc.

Usage

```
exportInstruments(rcon, ...)

## S3 method for class 'redcapDbConnection'
exportInstruments(rcon, ...)

## S3 method for class 'redcapApiConnection'
exportInstruments(
  rcon,
  ...,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

`rcon`	A REDCap connection object as generated by `redcapConnection`
`...`	Arguments to be passed to other methods.
`error_handling`	An option for how to handle errors returned by the API. see `redcap_error`

REDCap Version

6.5.0 +

5.8.2+

REDCap API Documentation

This function allows you to export a list of the data collection instruments for a project. This includes their unique instrument name as seen in the second column of the Data Dictionary, as well as each instrument's corresponding instrument label, which is seen on a project's left-hand menu when entering data. The instruments will be ordered according to their order in the project.

Known REDCap Limitations

None

Author(s)

Benjamin Nutter

References

Please refer to your institution's API documentation.

Additional details on API parameters are found on the package wiki at https://github.com/nutterb/redcapAPI/wiki/REDCap-API-Parameters

---

exportMappings *Exports the Event-Form Mappings for a Project*

Description

Retrieve a data frame giving the events-form mapping for a project.

Usage

```
exportMappings(rcon, arms, ...)

## S3 method for class 'redcapDbConnection'
exportMappings(rcon, arms, ...)

## S3 method for class 'redcapApiConnection'
exportMappings(
  rcon,
  arms = NULL,
  ...,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection.
arms	A vector of arm numbers that you wish to pull events for (by default, all events are pulled)
...	Arguments to be passed to other methods
error_handling	An option for how to handle errors returned by the API. see redcap_error

Details

The data frame that is returned shows the arm number, unique event name, and forms mapped in a project.

When this function is called for a classic project, a character string is returned giving the API error message, '400: You cannot export form-event mappings for classic projects' but without casting an error in R. This is by design and allows more flexible error checks in certain functions.

REDCap API Documentation

This function allows you to export the instrument-event mappings for a project (i.e., how the data collection instruments are designated for certain events in a longitudinal project).

NOTE: this only works for longitudinal projects

REDCap Version

5.8.2+ (and earlier, but we don't know how much earlier)

Known REDCap Limitations

None

Author(s)

Benjamin Nutter

References

Please refer to your institution's API documentation.

Additional details on API parameters are found on the package wiki at https://github.com/nutterb/redcapAPI/wiki/REDCap-API-Parameters

---

exportMetaData *Export Meta Data from a REDCap Database*

Description

Retrieves the meta data for a REDcap database, including field names, labels, types, formulas, etc. This file can be used to parse levels of factors, apply labels, and other data management tasks once the data are retrieved

Usage

```
exportMetaData(rcon, ...)

## S3 method for class 'redcapDbConnection'
exportMetaData(rcon, ...)

## S3 method for class 'redcapApiConnection'
exportMetaData(
  rcon,
  fields = NULL,
  forms = NULL,
  error_handling = getOption("redcap_error_handling"),
  ...,
  drop_utf8 = FALSE
)
```

Arguments

`rcon`	A REDCap connection object as generated by `redcapConnection`.
`...`	Arguments to be passed to other methods.
`fields`	A character vector of field names for which the metadata is to be retrieved.
`forms`	A character vector of forms for which the metadata is to be retrieved. Note that if a form name is given, all of the fields on that form will be returned, regardless of whether it is included in `fields` or not. Be careful to use the form names in the second column of the data dictionary, and not the display names shown on the webpage.
`error_handling`	An option for how to handle errors returned by the API. see `redcap_error`
`drop_utf8`	`logical(1)`. In some cases, UTF-8 characters can pose problems for exporting the data dictionary. Set this to `TRUE` to replace any UTF-8 characters with empty characters.

Details

A record of this export is placed in the REDCap logging page, but the file that is exported is not stored in the database.

REDCap API Documentation

This function allows you to export the metadata for a project

REDCap Version

5.8.2+ (and earlier, but we don't know how much earlier)

Known REDCap Limitations

The API doesn't respond to the `fields` and `forms` arguments. It always returns the full data dictionary.

Author(s)

Jeffrey Horner

`exportNextRecordName` *Generate Next Record Name from a REDCap Database*

Description

To be used by projects with record auto-numbering enabled, this method exports the next potential record ID for a project.

Usage

```
exportNextRecordName(rcon, ...)

## S3 method for class 'redcapDbConnection'
exportNextRecordName(rcon, ...)

## S3 method for class 'redcapApiConnection'
exportNextRecordName(
  rcon,
  ...,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection
...	Arguments to be passed to other methods.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Details

It generates the next record name by determining the current maximum numerical record ID and then incrementing it by one.

Value

Returns the maximum integer record ID + 1.

REDCap API Documentation

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

REDCap Version

8.1.8+

Author(s)

Xuefei Jia

References

Please refer to your institution's API documentation.

Additional details on API parameters are found on the package wiki at https://github.com/nutterb/redcapAPI/wiki/REDCap-API-Parameters

exportPdf *Export PDF file of Data Collection Instruments (either as blank or with data)*

Description

This function allows you to download PDF files of data collection instruments. The download may be with or without collected data, and may return a single record, multiple records, or all records.

Usage

```
exportPdf(
  rcon,
  dir,
  filename = "redcap_forms_download",
  record = NULL,
  events = NULL,
  instruments = NULL,
  all_records = FALSE,
  ...
)

## S3 method for class 'redcapDbConnection'
exportPdf(
  rcon,
  dir,
  filename = "redcap_forms_download",
  record = NULL,
  events = NULL,
  instruments = NULL,
  all_records = FALSE,
  ...
)

## S3 method for class 'redcapApiConnection'
exportPdf(
  rcon,
  dir,
  filename = "redcap_forms_download",
  record = NULL,
  events = NULL,
  instruments = NULL,
  all_records = FALSE,
  ...,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection.
dir	The directory into which the file should be saved.
filename	character(1). The base of the file name. If record = NULL, it will be appended with "_blank.pdf". If record has a value, it will be appended with "_record_[record id].pdf"
record	The record id for which forms should be downloaded. May only have length 1.
events	The events for which forms should be downloaded
instruments	The instruments for which forms should be downloaded
all_records	Logical. If TRUE forms for all records are downloaded. When TRUE, this overrides the records argument.
...	Arguments to be passed to other methods.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Details

This function mimics the behavior of "Download PDF of Instruments" button on the REDCap user interface.

Value

Creates a PDF file that is saved to the directory specified by the user.

REDCap API Documentation

This function allows you to export a PDF file for any of the following: 1) a single data collection instrument (blank), 2) all instruments (blank), 3) a single instrument (with data from a single record), 4) all instruments (with data from a single record), or 5) all instruments (with data from ALL records). This is the exact same PDF file that is downloadable from a project's data entry form in the web interface, and additionally, the user's privileges with regard to data exports will be applied here just like they are when downloading the PDF in the web interface (e.g., if they have de-identified data export rights, then it will remove data from certain fields in the PDF). If the user has "No Access" data export rights, they will not be able to use this method, and an error will be returned.

REDCap Version

6.5.0+

Known REDCap Limitations

None

Author(s)

Benjamin Nutter

exportProjectInformation *Exports the Project Information*

Description

Retrieve a data frame with the project information.

Usage

```
exportProjectInformation(rcon, ...)

## S3 method for class 'redcapDbConnection'
exportProjectInformation(rcon, ...)

## S3 method for class 'redcapApiConnection'
exportProjectInformation(
  rcon,
  ...,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection
...	Arguments to be passed to other methods.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Details

If this function is used in a version of REDCap that does not support the Export Version Number function, the character string 'Version Unknown' is returned.

REDCap API Documentation (6.5.0)

This function allows you to export some of the basic attributes of a given REDCap project, such as the project's title, if it is longitudinal, if surveys are enabled, the time the project was created and moved to production, etc.

REDCap Version

6.5.0 (Perhaps earlier)

Known REDCap Limitations

None

Author(s)

Benjamin Nutter

References

Please refer to your institution's API documentation.

Additional details on API parameters are found on the package wiki at https://github.com/nutterb/redcapAPI/wiki/REDCap-API-Parameters

---

`exportRecords` *Export Records from a REDCap Database*

Description

Exports records from a REDCap Database, allowing for subsets of subjects, fields, records, and events.

Usage

```
exportRecords(
  rcon,
  factors = TRUE,
  fields = NULL,
  forms = NULL,
  records = NULL,
  events = NULL,
  labels = TRUE,
  dates = TRUE,
  survey = TRUE,
  dag = TRUE,
  checkboxLabels = FALSE,
  colClasses = NA,
  ...
)

## S3 method for class 'redcapDbConnection'
exportRecords(
  rcon,
  factors = TRUE,
  fields = NULL,
  forms = NULL,
  records = NULL,
  events = NULL,
  labels = TRUE,
  dates = TRUE,
  survey = TRUE,
```

```
  dag = TRUE,
  checkboxLabels = FALSE,
  colClasses = NA,
  ...
)

## S3 method for class 'redcapApiConnection'
exportRecords(
  rcon,
  factors = TRUE,
  fields = NULL,
  forms = NULL,
  records = NULL,
  events = NULL,
  labels = TRUE,
  dates = TRUE,
  survey = TRUE,
  dag = TRUE,
  checkboxLabels = FALSE,
  colClasses = NA,
  ...,
  batch.size = -1,
  bundle = getOption("redcap_bundle"),
  error_handling = getOption("redcap_error_handling"),
  form_complete_auto = TRUE
)

exportRecords_offline(
  dataFile,
  metaDataFile,
  factors = TRUE,
  fields = NULL,
  forms = NULL,
  labels = TRUE,
  dates = TRUE,
  checkboxLabels = FALSE,
  colClasses = NA,
  ...,
  meta_data
)
```

Arguments

rcon	A REDCap connection object as created by redcapConnection.
factors	Logical. Determines if categorical data from the database is returned as numeric codes or labelled factors. See 'Checkbox Variables' for more on how this interacts with the checkboxLabels argument.
fields	A character vector of fields to be returned. If NULL, all fields are returned.

forms	A character vector of forms to be returned. If NULL, all forms are returned.
records	A vector of study id's to be returned. If NULL, all subjects are returned.
events	A character vector of events to be returned from a longitudinal database. If NULL, all events are returned.
labels	Logical. Determines if the variable labels are applied to the data frame.
dates	Logical. Determines if date variables are converted to POSIXct format during the download.
survey	specifies whether or not to export the survey identifier field (e.g., "redcap_survey_identifier") or survey timestamp fields (e.g., form_name+"_timestamp") when surveys are utilized in the project. If you do not pass in this flag, it will default to "false". If set to "true", it will return the redcap_survey_identifier field and also the survey timestamp field for a particular survey when at least one field from that survey is being exported. NOTE: If the survey identifier field or survey timestamp fields are imported via API data import, they will simply be ignored since they are not real fields in the project but rather are pseudo-fields.
dag	specifies whether or not to export the "redcap_data_access_group" field when data access groups are utilized in the project. If you do not pass in this flag, it will default to "false". NOTE: 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.
checkboxLabels	Logical. Determines the format of labels in checkbox variables. If FALSE labels are applies as "Unchecked"/"Checked". If TRUE, they are applied as ""/"[field_label]" where [field_label] is the label assigned to the level in the data dictionary. This option is only available after REDCap version 6.0. See Checkbox Variables for more on how this interacts with the factors argument.
colClasses	A (named) vector of colum classes passed to read.csv calls. Useful to force the interpretation of a column in a specific type and avoid an unexpected recast.
...	Additional arguments to be passed between methods.
batch.size	Integer. Specifies the number of subjects to be included in each batch of a batched export. Non-positive numbers export the entire project in a single batch. Batching the export may be beneficial to prevent tying up smaller servers. See details for more explanation.
bundle	A redcapBundle object as created by exportBundle.
error_handling	An option for how to handle errors returned by the API. see redcap_error
form_complete_auto	logical(1). When TRUE (default), the [form]_complete fields for any form from which at least one variable is requested will automatically be retrieved. When FALSE, these fields must be explicitly requested.
dataFile	For the offline version, a character string giving the location of the dataset downloaded from REDCap. Note that this should be the raw (unlabeled) data set.
metaDataFile	A text string giving the location of the data dictionary downloaded from REDCap.
meta_data	Deprecated version of metaDataFile

Details

A record of exports through the API is recorded in the Logging section of the project.

The 'offline' version of the function operates on the raw (unlabeled) data file downloaded from REDCap along with the data dictionary. This is made available for instances where the API can not be accessed for some reason (such as waiting for API approval from the REDCap administrator).

It is unnecessary to include "redcap_event_name" in the fields argument. This field is automatically exported for any longitudinal database. If the user does include it in the fields argument, it is removed quietly in the parameter checks.

A 'batched' export is one where the export is performed over a series of API calls rather than one large call. For large projects on small servers, this may prevent a single user from tying up the server and forcing others to wait on a larger job. The batched export is performed by first calling the API to export the subject identifier field (the first field in the meta data). The unique ID's are then assigned a batch number with no more than `batch.size` ID's in any single batch. The batches are exported from the API and stacked together.

In longitudinal projects, `batch.size` may not necessarily be the number of records exported in each batch. If `batch.size` is 10 and there are four records per patient, each batch will consist of 40 records. Thus, if you are concerned about tying up the server with a large, longitudinal project, it would be prudent to use a smaller batch size.

Checkbox Variables

There are four ways the data from checkbox variables may be represented depending on the values of `factors` and `checkboxLabels`. The most common are the first and third rows of the table below. When `checkboxLabels = TRUE`, either the coded value or the labelled value is returned if the box is checked, or an empty string if it is not.

factors	checkboxLabels	Output
FALSE	FALSE	0 / 1
FALSE	TRUE	"" / value
TRUE	FALSE	Unchecked / Checked
TRUE	TRUE	"" / label

REDCap API Documentation (6.5.0)

This function allows you to export a set of records for a project

Note about export rights (6.0.0+): Please be aware that Data Export user rights will be applied to this API request. For example, if you have "No Access" data export rights in the project, then the API data export will fail and return an error. And if you have "De-Identified" or "Remove all tagged Identifier fields" data export rights, then some data fields *might* be removed and filtered out of the data set returned from the API. To make sure that no data is unnecessarily filtered out of your API request, you should have "Full Data Set" export rights in the project.

REDCap Version

5.8.2 (Perhaps earlier)

Known REDCap Limitations

None

Deidentified Batched Calls

Batched calls to the API are not a feature of the REDCap API, but may be imposed by making multiple calls to the API. The process of batching the export requires that an initial call be made to the API to retrieve only the record IDs. The list of IDs is then broken into chunks, each about the size of `batch.size`. The batched calls then force the `records` argument in each call.

When a user's permissions require a de-identified data export, a batched call should be expected to fail. This is because, upon export, REDCap will hash the identifiers. When R attempts to pass the hashed identifiers back to REDCap, REDCap will try to match the hashed identifiers to the unhashed identifiers in the database. No matches will be found, and the export will fail.

Users who are exporting de-identified data will have to settle for using unbatched calls to the API (ie, `batch.size = -1`)

Author(s)

Jeffrey Horner

`exportReports` *Export Reports from a REDCap Database*

Description

Exports reports from a REDCap Database and formats data if requested

Usage

```
exportReports(
  rcon,
  report_id,
  factors = TRUE,
  labels = TRUE,
  dates = TRUE,
  checkboxLabels = FALSE,
  ...
)

## S3 method for class 'redcapDbConnection'
exportReports(
  rcon,
  report_id,
  factors = TRUE,
  labels = TRUE,
  dates = TRUE,
```

```
  checkboxLabels = FALSE,
  ...
)

## S3 method for class 'redcapApiConnection'
exportReports(
  rcon,
  report_id,
  factors = TRUE,
  labels = TRUE,
  dates = TRUE,
  checkboxLabels = FALSE,
  ...,
  bundle = getOption("redcap_bundle"),
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as created by redcapConnection.
report_id	Integer. Gives the report id of the desired report. This is located on the Report Builder page of the user interface on REDCap.
factors	Logical. Determines if categorical data from the database is returned as numeric codes or labelled factors.
labels	Logical. Determines if the variable labels are applied to the data frame.
dates	Logical. Determines if date variables are converted to POSIXlt format during the download.
checkboxLabels	Logical. Determines the format of labels in checkbox variables. If FALSE labels are applies as "Unchecked"/"Checked". If TRUE, they are applied as ""/"[field_labe]" where [field_label] is the label assigned to the level in the data dictionary. This option is only available after REDCap version 6.0.
...	Additional arguments to be passed between methods.
bundle	A redcapBundle object as created by exportBundle.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Details

A record of exports through the API is recorded in the Logging section of the project.

Reports are exported based on their id number, which can be looked up in the Reports page of a project

REDCap API Documentation (6.5.0)

This function allows you to export the data set of a report created on a project's "Data Exports, Reports, and Stats" page.

Note about export rights (6.0.0+): Please be aware that Data Export user rights will be applied to this API request. For example, if you have "No Access" data export rights in the project, then the API report export will fail and return an error. And if you have "De-Identified" or "Remove all tagged Identifier fields" data export rights, then some data fields *might* be removed and filtered out of the data set returned from the API. To make sure that no data is unnecessarily filtered out of your API request, you should have "Full Data Set" export rights in the project.

REDCap Version

6.0.0+

Known REDCap Limitations

None

Author(s)

Benjamin Nutter

---

exportSurveyParticipants

Exports details of participants for a given survey

Description

Retrieving dataframe of survey participants managed in REDCap

Usage

```
exportSurveyParticipants(rcon, instrument, event, ...)

## S3 method for class 'redcapDbConnection'
exportSurveyParticipants(rcon, instrument, event, ...)

## S3 method for class 'redcapApiConnection'
exportSurveyParticipants(
  rcon,
  instrument,
  event,
  ...,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection.
instrument	A string type holding the name of "instrument" or survey the participants are managed under
event	A string type holding the name of the event that the instrument belongs to
...	additional arguments to pass to other methods.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Details

REDCap allows the user to manage a list of participants (if they are known) for each survey. This function pulls this information into a dataframe. The contents of the dataframe are the contact fields of the participants, the link to the survey for that participant if the participant hasn't completed the survey yet (otherwise, NA), the participants record id, and other information.

Author(s)

Paddy Tobias

---

exportUsers *Export the Users for a Project*

Description

Retrieve a data frame giving the users, expiration dates, and data access privileges for each user.

Usage

```
exportUsers(rcon, ...)

## S3 method for class 'redcapDbConnection'
exportUsers(rcon, dates = TRUE, labels = TRUE, ...)

## S3 method for class 'redcapApiConnection'
exportUsers(
  rcon,
  dates = TRUE,
  labels = TRUE,
  ...,
  bundle = getOption("redcap_bundle"),
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection.
...	Arguments to be passed to other methods.
dates	Logical. Indicates if the expiration date is converted to a POSIXct object.
labels	Logical. Indicates if the data export and form access rights are converted to factor objects.
bundle	A redcap_bundle object.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Details

For some reason I have yet to identify, some User Tables do not export correctly. In some situations, the fields are all shifted one column to the left and the form names are not always exported. This seems to be more common in projects still in Development mode. I have seen one instance of a project in Production where one user had one more column given than any other user. If you notice this behavior, please report it to me as it may help me narrow down the source of the problem

Value

Returns a data frame. The number of columns in the data frame will depend on your version of REDCap.

- username User name
- email The user's e-mail address
- firstname The user's first name
- lastname The user's last name
- expiration The expiration date of the user's access to the project
- data_access_group The data access group the user is assigned to
- data_export The user's data export rights. 0=no access, 2=De-Identified, 1=Full Data Set
- mobile_app (6.5.0+) Flag for if the user has permissions for the mobile application
- mobile_app_download_data (6.5.0+) Flag for if the user may download data from the mobile app

The data frame will have one additional column for each form giving the user's form-level permissions in the project. 0=no access, 2=read only, 1=view records/responses and edit records (survey responses are read-only), 3 = edit survey responses

REDCap API Documentation (6.5.0)

This function allows you to export the users for a project

REDCap Version

5.8.2 (Perhaps earlier)

Known REDCap Limitations

None

Author(s)

Benjamin Nutter

References

Please refer to your institution's API documentation.

Additional details on API parameters are found on the package wiki at https://github.com/nutterb/redcapAPI/wiki/REDCap-API-Parameters

---

exportVersion *Exports the REDCap Version Number*

Description

Version numbers are returned as a character string. This feature is available for REDCap 6.0.0 and higher.

Usage

```
exportVersion(rcon, ...)

## S3 method for class 'redcapDbConnection'
exportVersion(rcon, ...)

## S3 method for class 'redcapApiConnection'
exportVersion(rcon, ..., error_handling = getOption("redcap_error_handling"))
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection
...	Arguments to be passed to other methods.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Details

If this function is used in a version of REDCap that does not support the Export Version Number function, the character string '5.12.2' is returned. This is done solely for the convenience of always returning a value that can be compared against other versions.

REDCap API Documentation (6.5.0)

This method returns the current REDCap version number as plain text (e.g., 4.13.18, 5.12.2, 6.0.0).

REDCap Version

6.0.0+

Known REDCap Limitations

None

Author(s)

Benjamin Nutter

References

Please refer to your institution's API documentation.

Additional details on API parameters are found on the package wiki at https://github.com/nutterb/redcapAPI/wiki/REDCap-API-Parameters

---

Extraction *Extraction and Assignment for* redcapFactors

Description

Extract elements and make assignments to redcapFactors

Usage

```
## S3 method for class 'redcapFactor'
x[..., drop = FALSE]

## S3 method for class 'redcapFactor'
print(x, ...)
```

Arguments

x	an object of class redcapFactor
...	additional arguments to pass to other methods
drop	logical. If TRUE, unused levels are dropped.

`fieldToVar` *Convert a REDCap Data Field to an R Vector*

Description

Converts a field exported from REDCap into a valid R vector

Usage

```
fieldToVar(
  records,
  meta_data,
  factors = TRUE,
  dates = TRUE,
  checkboxLabels = FALSE
)
```

Arguments

`records`	A data frame of records returned by `exportRecords` or `exportReports`
`meta_data`	A data frame giving the data dictionary, as returned by `exportMetaData`
`factors`	Logical, determines if checkbox, radio button, dropdown and yesno variables are converted to factors
`dates`	Logical, determines if date variables are converted to POSIXct format
`checkboxLabels`	Logical, determines if checkbox variables are labeled as "Checked" or using the checkbox label. Only applicable when `factors` = `TRUE`

Details

This function is called internally by `exportRecords` and `exportReports`. it is not available to the user.

Author(s)

Jeffrey Horner

genericApiCall *Generic Interface the REDCap API.*

Description

Permits users to make generic calls to the REDCap API. This allows use of API methods that do not yet have dedicated support.

Usage

```
genericApiCall(
  rcon,
  content,
  make_data_frame = TRUE,
  colClasses = NA,
  returnFormat = "csv",
  ...
)

## S3 method for class 'redcapDbConnection'
genericApiCall(
  rcon,
  content,
  make_data_frame = TRUE,
  colClasses = NA,
  returnFormat = "csv",
  ...
)

## S3 method for class 'redcapApiConnection'
genericApiCall(
  rcon,
  content,
  make_data_frame = TRUE,
  colClasses = NA,
  returnFormat = "csv",
  ...,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection.
content	character(1) The content argument for the API call.
make_data_frame	logical(1). When TRUE, an attempt is made to coerce the output to a data frame with read.csv. Otherwise, it is returned as a character vector.

colClasses	A named list of column names and classes to apply via read.csv
returnFormat	character(1) The format for the return. Defaults to "csv".
...	Additional named arguments giving arguments to the API method.
error_handling	An option for how to handle errors returned by the API. see redcap_error

---

importArms *Import Study Arm Names*

Description

Import Arms into a project or to rename existing Arms in a project. You may use the parameter override = TRUE as a 'delete all + import' action in order to erase all existing Arms in the project while importing new Arms. Notice: Because of the 'override' parameter's destructive nature, this method may only use override = TRUE for projects in Development status.

Usage

```
importArms(rcon, arms_data, override = FALSE, ...)

## S3 method for class 'redcapDbConnection'
importArms(rcon, arms_data, override = FALSE, ...)

## S3 method for class 'redcapApiConnection'
importArms(
  rcon,
  arms_data,
  override = FALSE,
  ...,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection.
arms_data	A data.frame with two columns. The first column is an integer-like value with the name arm_num. The second is a character value with the name name.
override	logical(1). When TRUE, the action is to delete all of the arms in the project and import the contents of arms_data. The default setting is FALSE, which only allows arms to be renamed or added.
...	additional arguments to pass to other methods.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Value

No value is returned.

REDCap API Documentation

This method allows you to import Arms into a project or to rename existing Arms in a project. You may use the parameter override=1 as a 'delete all + import' action in order to erase all existing Arms in the project while importing new Arms. Notice: Because of the 'override' parameter's destructive nature, this method may only use override=1 for projects in Development status.

NOTE: This only works for longitudinal projects.

REDCap Version

At least 8.1.17+ (and likely some earlier versions)

References

Please refer to your institution's API documentation.

---

importFiles *Imports a File to REDCap to Attach to a Record*

Description

A single file may be attached to a single record. The behavior of this function is consistent with the behavior of the API, which only allows one file to be uploaded at a time

Usage

```
importFiles(
  rcon,
  file,
  record,
  field,
  event,
  overwrite = TRUE,
  ...,
  bundle = NULL,
  repeat_instance = NULL
)

## S3 method for class 'redcapDbConnection'
importFiles(
  rcon,
  file,
  record,
  field,
  event,
  overwrite = TRUE,
  ...,
```

```
  bundle = NULL,
  repeat_instance = NULL
)

## S3 method for class 'redcapApiConnection'
importFiles(
  rcon,
  file,
  record,
  field,
  event = NULL,
  overwrite = TRUE,
  repeat_instance = NULL,
  ...,
  bundle = NULL,
  error_handling = getOption("redcap_error_handling")
)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection
file	Character string giving the file path to the file to be imported.
record	The record ID in which the desired file is stored. Must be length 1.
field	The field name in which the file is stored. Must be length 1.
event	The event name for the file. Must be length 1. This applies only to longitudinal projects. If the event is not supplied for a longitudinal project, the API will return an error
overwrite	Logical. When FALSE, the function checks if a file already exists for that record. If a file exists, the function terminates to prevent overwriting. When TRUE, no additional check is performed.
...	Arguments to be passed to other methods
bundle	A redcapBundle object as created by exportBundle.
repeat_instance	The repeat instance number of the repeating event or the repeating instrument. When available in your instance of REDCap, and passed as NULL, the API will assume a value of 1.
error_handling	An option for how to handle errors returned by the API. see redcap_error

Details

The function may only import a single file

Author(s)

Benjamin Nutter

importRecords *Import Records to a REDCap Database*

Description

Imports records from a data.frame to a REDCap Database

Usage

```
importRecords(
  rcon,
  data,
  overwriteBehavior = c("normal", "overwrite"),
  returnContent = c("count", "ids", "nothing"),
  returnData = FALSE,
  logfile = "",
  ...
)

## S3 method for class 'redcapDbConnection'
importRecords(
  rcon,
  data,
  overwriteBehavior = c("normal", "overwrite"),
  returnContent = c("count", "ids", "nothing"),
  returnData = FALSE,
  logfile = "",
  ...
)

## S3 method for class 'redcapApiConnection'
importRecords(
  rcon,
  data,
  overwriteBehavior = c("normal", "overwrite"),
  returnContent = c("count", "ids", "nothing"),
  returnData = FALSE,
  logfile = "",
  ...,
  bundle = NULL,
  batch.size = -1
)
```

Arguments

rcon	A REDCap connection object as created by redcapConnection.
data	A data.frame to be imported to the REDCap project.

`overwriteBehavior`
: Character string. 'normal' prevents blank fields from overwriting populated fields. 'overwrite' causes blanks to overwrite data in the REDCap database.

`returnContent`
: Character string. 'count' returns the number of records imported; 'ids' returns the record ids that are imported; 'nothing' returns no message.

`returnData`
: Logical. Prevents the REDCap import and instead returns the data frame that would have been given for import. This is sometimes helpful if the API import fails without providing an informative message. The data frame can be written to a csv and uploaded using the interactive tools to troubleshoot the problem. Please shoot me an e-mail if you find errors I havne't accounted for.

`logfile`
: An optional filepath (preferably .txt) in which to print the log of errors and warnings about the data. If `""`, the log is printed to the console.

`...`
: Arguments to be passed to other methods.

`bundle`
: A `redcapBundle` object as created by `exportBundle`.

`batch.size`
: Specifies size of batches. A negative value indicates no batching.

Details

A record of imports through the API is recorded in the Logging section of the project.

`importRecords` prevents the most common import errors by testing the data before attempting the import. Namely

1. Check that all variables in `data` exist in the REDCap data dictionary.
2. Check that the study id variable exists
3. Force the study id variable to the first position in the data frame (with a warning)
4. Remove calculated fields (with a warning)
5. Verify that REDCap date fields are represented in the data frame as either character, POSIXct, or Date class objects.
6. Determine if values are within their specified validation limits.

See the documentation for `validateImport` for detailed explanations of the validation.

Author(s)

Benjamin Nutter
with thanks to Josh O'Brien and etb (see references)

References

See the REDCap API documentation at your institution's REDCap documentation.

See Also

`validateImport`

massert *Conduct Multiple Assertions*

Description

This documentation attempts to describe arguments to make assertions on arguments. In order to prevent confusion, it is imperative to develop some terminology up front. We will use *function argument* to refer to an argument of the function for which we are conducting assertions. We will use *assertion argument* to refer to arguments to pass to the assertion function being applied to a function argument. Lastly, we will use *massert argument* to refer to arguments to `massert`

Usage

```
massert(formula, fun, ..., fixed = list())
```

Arguments

`formula`	A one sided formula naming the arguments on which the assertion will be performed.
`fun`	An assertion function to perform.
`...`	Additional lists. Each argument provided is a named list of `assertion arguments`. The name of each element in a list should match the name of a `function argument`. `lower = list(var1 = 0, var2 = 10)` sets the *assertion argument* `lower = 0` for *function argument* `var1`; and sets the *assertion argument* `lower = 10` for `function argument var2`. The *massert arguments* in `...` may themselves be named or unnamed.
`fixed`	A named list of arguments that are fixed across all assertions.

Details

Only one assert function may be utilized in each call to `massert`. This allows for all numeric variables to be checked in one call, all logical variables to be checked in a subsequent call, etc.

Author(s)

Benjamin Nutter

parseBranchingLogic *Parse Branching Logic*

Description

Branching logic from the REDCap Data Dictionary is parsed into R Code and returned as expressions. These can be evaluated if desired and allow the user to determine if missing values are truly missing or not required because the branching logic prevented the variable from being presented.

Usage

```
parseBranchingLogic(l)
```

Arguments

l	A vector of REDCap branching logic statements. These are usually passed as the vector `meta_data$branching_logic`.

Details

For a study, I was asked to identify which subjects had missing values so that remaining data could be collected. The initial pass of `is.na` produced a lot of subjects missing values where there was no need to collect data because they did not qualify for some variables in the branching logic. Parsing the logic allowed me to determine which values we expected to be missing and narrow the search to just those subjects with legitimately missing values.

Value

Returns a list of unevaluated expressions.

Author(s)

Benjamin Nutter

recodeCheck *Change labelling of* checkbox *variables*

Description

Rewrites the labelling of `checkbox` variables from Checked/Unchecked to Yes/No (or some other user-specified labelling).

Usage

```
recodeCheck(
  df,
  vars,
  old = c("Unchecked", "Checked"),
  new = c("No", "Yes"),
  reverse = FALSE
)
```

Arguments

df	A data frame, presumably retrieved from REDCap, though not a strict requirement.
vars	Optional character vector of variables to convert. If left missing, all of the variables in df that are identified as checkbox variables are relabelled. See 'Details' for more about identifying checkbox variables.
old	A character vector to be passed to factor. This indicates the levels to be replaced and their order.
new	A character vector of labels to replace the values in levels. The first value becomes the reference value.
reverse	For convenience, if the user would prefer to reverse the order of the elements in levels and labels, simply set this to TRUE.

Details

checkbox variables are *not* identified using the metadata from the REDCap database. Instead, variables are scanned, and those variables in which every value is in levels are assumed to be checkbox variables.

Realistically, this could be used to relabel any set of factors with identical labels, regardless of the data source. The number of labels is not limited, but levels and labels should have the same length.

The actual code to perform this is not particularly difficult (df[checkbox] <- lapply(df[checkbox], factor, levels=levels, labels=labels)), but checkbox variables are common enough in REDCap (and the Checked/Unchecked scheme so unpalatable) that a quick way to replace the labels was highly desirable

Author(s)

Benjamin Nutter

redcapAPI *Access data, meta data, and files from REDCap using the API*

Description

REDCap is a database development tool built on MySQL. Visit https://project-redcap.org for more information. REDCap provides an API through which data, the data dictionary, files, and project information can be accessed. The redcapAPI package facilitates the use of these functions and simplifies the work needed to prepare data for analysis.

Details

As much as possible, I've tried to adequately document redcapAPI. Some topics did not seem well-suited to documenting in the typical R help files. Additional tips and discussion are available at the package wiki at https://github.com/nutterb/redcapAPI/wiki. These topics include "Getting started with redcapAPI", "Setting Rights to Grant API Access", "Export data from REDCap" and a detailed description of the REDCap API parameters and how they are implemented in R. I expect most documentation improvements to be placed on the wiki.

Please refer to your institution's REDCap API documentation as a primary resource of what is available. Different versions of REDCap support different features–your REDCap API documentation will address the features specific to your version of REDCap.

redcapAPI wouldn't be possible without the efforts of Jeffrey Horner, Will Gray, and Jeremy Stephens at Vanderbilt University. Their work in developing the original redcap package. was invaluable in helping me understand the API. A few of their functions (redcapConnection, fieldToVar, exportMetaData, and exportRecords) are included in redcapAPI largely unaltered.

Many thanks also go to Will Beasley of University of Oklahoma for his development of the REDCapR package https://github.com/OuhscBbmc/REDCapR. Will introduced me to the httr package–which improved the use of messages from the API–and to the idea of batching API calls to reduce the likelihood of servers timing out.

redcapConnection *Connect to a REDCap Database*

Description

Creates an object of class redcapApiConnection for using the REDCap API [or a direct connection through an SQL server]

Usage

```
redcapConnection(
  url = getOption("redcap_api_url"),
  token,
  conn,
```

```
  project,
  config = httr::config()
)
```

Arguments

url	URL for a REDCap database API. Check your institution's REDCap documentation for this address. Either url or conn must be specified.
token	REDCap API token
conn	The database connection to be used. If used, project must also be used.
project	The project ID in the REDCap tables.
config	A list to be passed to httr::POST. This allows the user to set additional configurations for the API calls, such as certificates, ssl version, etc. For the majority of users, this does not need to be altered. See Details for more about this argument's purpose and the redcapAPI wiki for specifics on its use.

Details

For convenience, you may consider using options(redcap_api_url=[your URL here]) in your RProfile. To obtain an API token for a project, do the following:
Enter the 'User Right' section of a project
Select a user
Check the box for 'API Data Export' or 'API Data Import,' as appropriate. A full tutorial on configuring REDCap to use the API can be found at https://github.com/nutterb/redcapAPI/wiki

Tokens are specific to a project, and a token must be created for each project for which you wish to use the API.

The config argument is passed to the httr::POST argument of the same name. The most likely reason for using this argument is that the certificate files bundled in httr have fallen out of date. Hadley Wickham is pretty good about keeping those certificates up to date, so most of the time this problem can be resolved by updating httr to the most recent version. If that doesn't work, a certificate file can be manually passed via the config argument. The redcapAPI wiki has a more detailed tutorial on how to find and pass an SSL certificate to the API call (https://github.com/nutterb/redcapAPI/wiki/Manually-Setting-an-SSL-Certificate-File).

Additional Curl option can be set in the config argument. See the documentation for httr::config and httr:httr_options for more Curl options.

Author(s)

Jeffrey Horner

Examples

```
## Not run:
rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])

options(redcap_api_url=[YOUR_REDCAP_URL])
rcon <- redcapConnection(token=[API_TOKEN])
```

```
exportRecords(rcon)

## End(Not run)
```

`redcapFactorFlip` *Convert REDCap factors between labelled and coded*

Description

Factors exported from REDCap can be toggled between the coded and labelled values with the use of the attributes assigned to the factors during export.

Usage

```
redcapFactorFlip(v)
```

Arguments

`v`	A factor exported from REDCap. The REDCap type may be radio, dropdown, check, yesno, etc.

Details

Each factor type variable in REDCap is given the attributes `redcapLabels` and `redcapLevels`. With these attached to the vector, switching between the coded and labelled values can be done with ease. This may be helpful when the coded value has importance, such as 0/1 for death, or if a yes is worth 6 points (instead of 1).

Author(s)

Benjamin Nutter

`redcap_error` *Handle Errors from the REDCap API*

Description

Determine the proper way to handle errors returned from the API. Not all errors should be fatal. See Details for more

Usage

```
redcap_error(x, error_handling)
```

Arguments

x	Object returned by POST.
error_handling	Direction for how to handle errors. May be either "null" or "error". See Details.

Details

Maintaining consistent functionality for all types of REDCap projects requires that errors be handled delicately. It is not always desirable for an error from the API to terminate the program. One example of such a case is when a user executes the exportEvents function for a classic project; doing so returns an error from the API that events cannot be exported for classic projects. In REDCap versions earlier than 6.5.0, there is no way to determine if a project is classic or longitudinal without attempting to export the events.

For this reason, it is often preferable to have these kinds of errors return NULL so that the program doesn't crash if it doesn't need to (one such instance where it doesn't need to crash is when exportEvents is called within exportRecords; the events argument is irrelevant to a classic project and the error can safely be ignored.

The other common type of error that does not need to be fatal is when a redcapAPI method is sent to a REDCap instance that does not support the method. For example, the exportVersion method is not supported in REDCap instances earlier than 6.0.0. In these cases, we may prefer not to cast a hard error.

These two types of errors may be handled in one of two ways. When the error handler is set to "null", a NULL is returned. When the error handler is set to "error", the error is returned. The option is set globally using options(redcap_error_handler = "null") and is set to "null" by default.

Handled Errors

Only the errors listed below are handled. All others throw a hard error.

"ERROR: The value of the parameter \"content\" is not valid"
"ERROR: You cannot export arms for classic projects"
"ERROR: You cannot export events for classic projects"

Author(s)

Benjamin Nutter

---

syncUnderscoreCodings *Sychronize coding of checkbox variables between meta data and records field names.*

Description

Due to a bug in the REDCap export module, underscores in checkbox codings are not retained in the suffixes of the field names in the exported records. For example, if variable `chk` is a checkbox with a coding 'a_b, A and B', the field name in the data export becomes `chk___ab`. The loss of the underscore causes `fieldToVar` to fail as it can't match variable names to the meta data. `syncUnderscoreCodings` rectifies this problem by searching the suffixes and meta data for underscores. If a discrepancy is found, the underscores are removed from the metadata codings, restoring harmony to the universe. This bug was fixed in REDCap version 5.5.21 and this function does not apply to that and later versions.

Usage

```
syncUnderscoreCodings(records, meta_data, export = TRUE)
```

Arguments

`records`	The data frame object returned from the API export prior to applying factors, labels, and dates via the `fieldToVar` function.
`meta_data`	Metadata export from `exportMetaData`
`export`	Logical. Specifies if data are being synchronized for import or export

Details

`syncUnderscoreCodings` performs a series of evaluations. First, it determines if any underscores are found in the checkbox codings. If none are found, the function terminates without changing anything.

If the checkbox codings have underscores, the next evaluation is to determine if the variable names suffixes have matching underscores. If they do, then the function terminates with no changes to the meta data.

For data exports, if the prior two checks find underscores in the meta data and no underscores in the suffixes, the underscores are removed from the meta data and the new meta data returned.

For data imports, the meta data are not altered and the `checkbox_field_name_map` attribute is used to synchronize field names to the meta data and the expectations of REDCap (for import, REDCap expects the underscore codings to be used.

Backward Compatibility

In retrospect, we realize that the way `syncUnderscoreCodings` is written is backwards. We should have altered the field names in the records data frame. Any scripts that make use of `syncUnderscoreCodings` and were written prior to version 5.5.21 will fail because the underscores in the codings will now be present where they weren't before.

For backward compatibility of `redcapAPI`, we continue to alter the codings in the meta data. We do not anticipate many problems, as most people don't use underscores in the checkbox codings

If your scripts were written under REDCap 5.5.21 or higher, you will have no backward compatibility problems related to this issue.

Author(s)

Benjamin Nutter

`validateImport` *Validate Data Frames for Import*

Description

Validates the variables in a data frame prior to attempting an import to REDCap

Usage

```
validateImport(data, meta_data, logfile = "")
```

Arguments

`data`	Data frame being prepared for import to REDCap.
`meta_data`	REDCap database meta data.
`logfile`	A character string giving the filepath to which the results of the validation are printed. If `""`, the results are printed in the console.

Details

`validateImport` is called internally by `importRecords` and is not available to the user.

Each variable is validated by matching they type of variable with the type listed in the REDCap database.

Although the log messages will indicate a preference for dates to be in mm/dd/yyyy format, the function will accept mm/dd/yy, yyyy-mm-dd, yyyy/mm/dd, and yyyymmdd formats as well. When possible, pass dates as Date objects or POSIXct objects to avoid confusion. Dates are also compared to minimum and maximum values listed in the data dictionary. Records where a date is found out of range are allowed to import and a message is printed in the log.

For continuous/numeric variables, the values are checked against the minimum and maximum allowed in the data dictionary. Records where a value is found out of range are allowed to import and a message is printed in the log.

ZIP codes are tested to see if they fit either the 5 digit or 5 digit + 4 format. When these conditions are not met, the data point is deleted and a message printed in the log.

YesNo fields permit any of the values 'yes', 'no', '0', '1' to be imported to REDCap with 0=No, and 1=Yes. The values are converted to lower case for validation, so any combination of lower and upper case values will pass (ie, the data frame is not case-sensitive).

TrueFalse fields will accept 'TRUE', 'FALSE', 0, 1, and logical values and are also not case-sensitive.

Radio and dropdown fields may have either the coding in the data dictionary or the labels in the data dictionary. The validation will use the meta data to convert any matching values to the appropriate

coding before importing to REDCap. Values that cannot be reconciled are deleted with a message printed in the log. These variables are not case-sensitive.

Checkbox fields require a value of "Checked", "Unchecked", "0", or "1". These are currently case sensitive. Values that do not match these are deleted with a warning printed in the log.

Phone numbers are required to be 10 digit numbers. The phone number is broken into three parts: 1) a 3 digit area code, 2) a 3 digit exchange code, and 3) a 4 digit station code. The exchange code must start with a number from 2-9, followed by 0-8, and then any third digit. The exchange code starts with a number from 2-9, followed by any two digits. The station code is 4 digits with no restrictions.

E-mail addresses are considered valid when they have three parts. The first part comes before the @ symbol, and may be number of characters from a-z, A-Z, a period, underscore, percent, plus, or minus. The second part comes after the @, but before the period, and may consist of any number of letters, numbers, periods, or dashes. Finally, the string ends with a period then anywhere from 2 to 6 letters.

Author(s)

Benjamin Nutter

References

See the REDCap Help and FAQ page's section on 'Text Validation Types'

Index