Package 'redcapAPI'

December 18, 2017

Type Package

Title R Interface to REDCap

Version 1.3.7-2

Author Benjamin Nutter. Initiated by Jeffrey Horner and Will Gray with contributions from Jeremy Stephens, and Will Beasley

Maintainer Stephen Lane <lane.s@unimelb.edu.au>

Description Access data stored in REDCap databases using the Application Programming Interface (API). REDCap (Research Electronic Data CAPture) 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 chron, DBI, Hmisc (>= 3.14-6), httr, stringr

LazyLoad yes

URL https://github.com/nutterb/redcapAPI/wiki, https://github.com/nutterb/redcapAPI, http://project-redcap.org

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

RoxygenNote 6.0.1

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, proj = 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, proj = 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, proj = 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, weights = NULL, ...)
```

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.
proj	A redcapProjectInfo 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

---

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, field, event, ...,
  proj = NULL)
```

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
`proj`	A `redcapProject` object as created by `redcapProjectInfo`.

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

Examples

```
## Not run:
  > #*** Note: I cannot provide working examples without
  > #*** compromising security.  Instead, I will try to
  > #*** offer up sample code with the matching results
  >
  > #*** Create the connection object
  > rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])
  >
  >
  > #* Delete a file
  > deleteFiles(rcon, record=1, field="file_upload", event="event_1_arm_1")
  The file was successfully deleted
  >

## End(Not run)
```

`exportArms` *Exports the Arms for a Project*

Description

Retrieve a data frame giving the event IDs and event names of a project

Usage

```
exportArms(rcon, arms, ...)

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

## S3 method for class 'redcapApiConnection'
exportArms(rcon, arms, ...)
```

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). A bug exists in early versions of the API that causes all arms to be returned regardless of this argument. This bug was fixed in version 5.9.15
`...`	Arguments to be passed to other methods

Details

The data frame that is returned shows the arm number, and arm name.

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

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

Examples

```
## Not run:
> #*** Note: I cannot provide working examples without
> #*** compromising security.  Instead, I will try to
> #*** offer up sample code with the matching results
>
>
> #*** Create the connection object
> rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])
>
> exportArms(rcon)
arm_num             name
1       1            Arm 1
2       2 Experimental Arm
3      10   Normal Control

## End(Not run)
```

---

exportEvents *Export the Events for a Project*

Description

Retrieve a data frame giving the events, event names, and offsets for the events in a project

Usage

```
exportEvents(rcon, arms, ...)

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

## S3 method for class 'redcapApiConnection'
exportEvents(rcon, arms, ...)
```

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.

Details

The data frame that is returned shows the event name, arm number, days offset, minimum offset, maximum offset, and unique event name.

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

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

Examples

```
## Not run:
> #*** Note: I cannot provide working examples without
> #*** compromising security.  Instead, I will try to
> #*** offer up sample code with the matching results
>
> #*** Create the connection object
> rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])
>
> exportEvents(rcon)
event_name arm_num day_offset offset_min offset_max unique_event_name
1     Event 1       1          0          0          0     event_1_arm_1
2 Follow Up 1       1          1          0          0 follow_up_1_arm_1
3 Follow Up 2       1          2          0          0 follow_up_2_arm_1

## End(Not run)
```

---

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, ...,
  proj = NULL)

## S3 method for class 'redcapDbConnection'
exportFiles(rcon, record, field, event, dir,
  filePrefix = TRUE, ..., proj = NULL)

## S3 method for class 'redcapApiConnection'
exportFiles(rcon, record, field, event, dir,
  filePrefix = TRUE, ..., proj = NULL)
```

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
proj	A redcapProject object as created by redcapProjectInfo.

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

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

Examples

```
## Not run:
> #*** Note: I cannot provide working examples without
> #*** compromising security.  Instead, I will try to
> #*** offer up sample code with the matching results
```

```
>
> #*** Create the connection object
> rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])
>
>
> #* Export a single file
> exportFiles(rcon, record=1, field="file_upload", event="event_1_arm_1")
The file was saved to '1-event_1_arm_1-NewOutcomes.xlsx'
>
>
> #* Export all files in a project
> #* Although this example only shows one field for files, it could work with
> #* an arbitrary number of file upload fields
> library(reshape2)
> Data <- exportRecords(Data)
> (filesToExport <- melt(Data[, c("id", "redcap_event_name", "file_upload")],
                         c("id", "redcap_event_name")),
                         na.rm=TRUE)
id redcap_event_name    variable      value
1  1     event_1_arm_1 file_upload [document]
4  2     event_1_arm_1 file_upload [document]
>
> for(i in 1:nrow(filesToExport)){
+   exportFiles(rcon, record=filesToExport$id[i],
+               field=filesToExport$variable[i],
+               event=filesToExport$redcap_event_name[i])
+ }
The file was saved to '1-event_1_arm_1-NewOutcomes.xlsx'
The file was saved to '2-event_1_arm_1-Sunset2.JPG'

## End(Not run)
```

`exportInstruments` *Export Instrument names*

Description

Retrieve a data frame giving the instrument names and labels in a project.

Usage

```
exportInstruments(rcon, ...)

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

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

Arguments

`rcon`	A REDCap connection object as generated by `redcapConnection`
`...`	Arguments to be passed to other methods

Details

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.

This function was introduced to the API in version 5.9. An error will be returned if called to an instance earlier than 5.9.

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, ...)
```

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

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.

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

Examples

```
## Not run:
> #*** Note: I cannot provide working examples without
> #*** compromising security.  Instead, I will try to
> #*** offer up sample code with the matching results
>
>
> #*** Create the connection object
> rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])
>
> exportMappings(rcon)
arm_num        unique_event_name                form_name
1        1            event_1_arm_1              demographics
2        1            event_1_arm_1 all_the_different_options
3        1            event_1_arm_1        uploading_decimals
4        1            event_1_arm_1              calculations
5        1        follow_up_1_arm_1              calculations
6        1        follow_up_2_arm_1              calculations
7        2 experimental_inter_arm_2              demographics
8        2 experimental_inter_arm_2        uploading_decimals
9        2 experimental_follo_arm_2 all_the_different_options
10       2 experimental_follo_arm_2        uploading_decimals
11       2 experimental_follo_arm_2              calculations
12      10          baseline_arm_10              demographics
13      10          baseline_arm_10        uploading_decimals
>
> exportMappings(rcon, 1:2)
arm_num        unique_event_name                form_name
1        1            event_1_arm_1              demographics
2        1            event_1_arm_1 all_the_different_options
3        1            event_1_arm_1        uploading_decimals
4        1            event_1_arm_1              calculations
5        1        follow_up_1_arm_1              calculations
6        1        follow_up_2_arm_1              calculations
7        2 experimental_inter_arm_2              demographics
8        2 experimental_inter_arm_2        uploading_decimals
9        2 experimental_follo_arm_2 all_the_different_options
10       2 experimental_follo_arm_2        uploading_decimals
11       2 experimental_follo_arm_2              calculations

## End(Not run)
```

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, ...)
```

Arguments

rcon	A REDCap connection object as generated by redcapConnection.
...	Arguments to be passed to other methods.

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.

Author(s)

Jeffrey Horner

References

This functionality were originally developed by Jeffrey Horner in the redcap package. https://github.com/vubiostat/redcap

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

Examples

```
## Not run:
#*** Note: I cannot provide working examples without
#*** compromising security.  Instead, I will try to
#*** offer up sample code with the matching results

#*** Create the connection object
rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])

exportMetaData(rcon)
```

```
## End(Not run)
```

`exportRecords` *Export Records from a REDCap Database*

Description

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

Usage

```
queryRecords(rcon, fields = NULL, forms = NULL, records = NULL,
  events = NULL)

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, ...)

## 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, ..., batch.size = -1, proj = NULL,
  colClasses = NA)

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

Arguments

`rcon`	A REDCap connection object as created by `redcapConnection`.
`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.
`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 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_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.
colClasses	A (named) vector of colum classes passed to [utils::read.csv()]utils::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.
proj	A redcapProject object as created by redcapProjectInfo.
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.
meta_data	A text string giving the location of the data dictionary downloaded from REDCap.

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.

Author(s)

Jeffrey Horner

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

This functionality were originally developed by Jeffrey Horner in the redcap package. https://github.com/vubiostat/redcap

See also read_redcap_oneshot in the REDCapR package by Will Beasley. https://github.com/OuhscBbmc/REDCapR

Examples

```
## Not run:
> #*** Note: I cannot provide working examples
> #*** without compromising security.  Instead, I will try to
> #*** offer up sample code with the matching results
>
>
> #*** Create the connection object
> rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])
>
>
> #*** Export the full data set
> BMD <- exportRecords(rcon)
> head(BMD)
patient_id redcap_event_name      bmi patient_characteristics_complete
1          1       entry_arm_1 38.18765                                2
2          1  dxa_scan_1_arm_1       NA                               NA
3          1  dxa_scan_2_arm_1       NA                               NA
4          1  dxa_scan_3_arm_1       NA                               NA
5          2       entry_arm_1 24.40972                                2
6          2  dxa_scan_1_arm_1       NA                               NA
contact_date hip_left_bmd hip_left_tscore hip_right_bmd hip_right_tscore
1         <NA>           NA              NA            NA               NA
2   2013-06-12           NA              NA            NA               NA
3   2009-02-11           NA              NA            NA               NA
4   2011-02-26           NA              NA            NA               NA
5         <NA>           NA              NA            NA               NA
6   2010-11-06        0.697              -2            NA               NA
neck_left_bmd neck_left_tscore neck_right_bmd neck_right_tscore spine_bmd
1            NA              NA             NA                NA        NA
2         0.664            -2.0             NA                NA        NA
3         0.675            -1.9             NA                NA        NA
4         0.734            -1.5             NA                NA        NA
5            NA              NA             NA                NA        NA
6         0.521            -3.0             NA                NA     0.899
```

```
spine_tscore dxa_scan_summary_complete
1           NA                        NA
2           NA                         2
3           NA                         2
4           NA                         2
5           NA                        NA
6         -1.3                         2
>
>
>
> #*** Export only the patient_characteristics form
> BMD <- exportRecords(rcon, forms="patient_characteristics")
> head(BMD)
patient_id redcap_event_name      bmi patient_characteristics_complete
1          1       entry_arm_1 38.18765                               2
2          1  dxa_scan_1_arm_1       NA                              NA
3          1  dxa_scan_2_arm_1       NA                              NA
4          1  dxa_scan_3_arm_1       NA                              NA
5          2       entry_arm_1 24.40972                               2
6          2  dxa_scan_1_arm_1       NA                              NA
>
>
> #*** Export only the second scan
> BMD <- exportRecords(rcon, events="dxa_scan_2_arm_1", forms="dxa_scan_summary")
> head(BMD)
patient_id redcap_event_name contact_date hip_left_bmd hip_left_tscore
1          1  dxa_scan_2_arm_1   2009-02-11           NA              NA
2          2  dxa_scan_2_arm_1   2012-10-30        0.684            -2.1
3          3  dxa_scan_2_arm_1   2013-02-06        1.007             0.0
4          4  dxa_scan_2_arm_1   2007-09-20           NA              NA
5          5  dxa_scan_2_arm_1   2006-07-07           NA              NA
6          6  dxa_scan_2_arm_1   2006-10-25           NA              NA
hip_right_bmd hip_right_tscore neck_left_bmd neck_left_tscore neck_right_bmd
1            NA               NA         0.675             -1.9             NA
2            NA               NA         0.524             -2.9             NA
3            NA               NA         0.897             -1.0             NA
4            NA               NA         0.632             -2.0             NA
5            NA               NA         0.835             -0.1             NA
6            NA               NA            NA               NA           0.54
neck_right_tscore spine_bmd spine_tscore dxa_scan_summary_complete
1                NA        NA           NA                        2
2                NA     0.915         -1.2                        2
3                NA     1.109         -0.6                        2
4                NA     0.864         -1.7                        2
5                NA     0.869         -1.6                        2
6              -2.8     0.830         -2.0                        2
>
>
> #*** Retrieve the first scan for patients 38 and 103
> BMD <- exportRecords(rcon, records=c(38, 103),
                       forms="dxa_scan_summary", events="dxa_scan_1_arm_1")
> BMD
patient_id redcap_event_name contact_date hip_left_bmd hip_left_tscore
1          38  dxa_scan_1_arm_1   2008-05-07           NA              NA
2         103  dxa_scan_1_arm_1   2010-04-21        0.856            -1.2
hip_right_bmd hip_right_tscore neck_left_bmd neck_left_tscore neck_right_bmd
1            NA               NA         0.595             -2.3             NA
```

```
2            NA               NA         0.789             -1.8             NA
neck_right_tscore spine_bmd spine_tscore dxa_scan_summary_complete
1                NA     0.770         -2.5                         2
2                NA     1.023         -1.3                         2

## End(Not run)
```

`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, ..., proj = NULL)
```

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.
`proj`	A `redcapProject` object as created by `redcapProjectInfo`.

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

Author(s)

Benjamin Nutter

`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, date = TRUE, label = TRUE,
  ...)

## S3 method for class 'redcapApiConnection'
exportUsers(rcon, date = TRUE, label = TRUE,
  ...)
```

Arguments

`rcon`	A REDCap connection object as generated by `redcapConnection`.
`...`	Arguments to be passed to other methods.
`date`	Logical. Indicates if the expiration date is converted to a `POSIXct` object.
`label`	Logical. Indicates if the data export and form access rights are converted to factor objects.

Details

From the REDCap API Documentation:

Data Export: 0=no access, 2=De-Identified, 1=Full Data Set
Form Rights: 0=no access, 2=read only, 1=view records/responses and edit records (survey responses are read-only), 3 = edit survey responses

(NOTE: At this time, only a limited amount of rights-related info will be exported (expiration, data access group ID, data export rights, and form-level rights). However, more info about a user's rights will eventually be added to the Export Users API functionality in future versions of REDCap.)

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

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

Examples

```
## Not run:
> #*** Note: I cannot provide working examples without
> #*** compromising security.  Instead, I will try to
> #*** offer up sample code with the matching results
>
>
> #*** Create the connection object
> rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])
>
> exportUsers(rcon)
> #*** Note: I cannot provide working examples without
> #*** compromising security.  Instead, I will try to
> #*** offer up sample code with the matching results
>
>
> #*** Create the connection object
> rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])
>
> exportUsers(rcon)
username            email firstname lastname expiration data_access_group
1    user1 user1@domain.org     Name1 Surname1       <NA>                NA
2    user2             <NA>      <NA>     <NA>       <NA>                NA
3    user3 user3@domain.org     Name3 Surname3       <NA>                NA
4    user4             <NA>      <NA>     <NA>       <NA>                NA
5    user5 user5@domain.org     Name5 Surname5       <NA>                NA
6    user6             <NA>      <NA>     <NA>       <NA>                NA
7    user7 user6@domain.org     Name7 Surname7       <NA>                NA
data_export                 patient_characteristics
1 Full data set view records/responses and edit records
2 De-identified view records/responses and edit records
3 De-identified view records/responses and edit records
4 De-identified view records/responses and edit records
5 Full data set view records/responses and edit records
6 De-identified view records/responses and edit records
7 Full data set view records/responses and edit records
dxa_scan_summary
1 view records/responses and edit records
2 view records/responses and edit records
3 view records/responses and edit records
4 view records/responses and edit records
5 view records/responses and edit records
6 view records/responses and edit records
7 view records/responses and edit records

## End(Not run)
```

`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, ...)
```

Arguments

`rcon`	A REDCap connection object as generated by `redcapConnection`
`...`	Arguments to be passed to other methods.

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.

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

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

Description

Converts a field exported from REDCap into a valid R vector

Usage

```
fieldToVar(m, d, factors = TRUE, dates = TRUE, checkboxLabels = FALSE,
  vname)
```

Arguments

m	a metadata file, as returned by exportMetaData
d	A data files, as returned by exportRecords
factors	Logical. Determines if categorical factors from REDCap are returned with their numeric codes or as labelled factors.
dates	Logical. Determines if date variables from REDCap are converted to POSIXct format. The API returns dates as character strings by default in YYYY-MM-DD format.
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 only applies when factors=TRUE and only to REDCap versions 6.0 and higher
vname	The variable name being converted. This is used only when checkboxLabels=TRUE in order to extract the checkbox label.

Details

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

Author(s)

Jeffrey Horner

---

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, ...,
  proj = NULL)

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

## S3 method for class 'redcapApiConnection'
importFiles(rcon, file, record, field, event,
  overwrite = TRUE, ..., proj = NULL)
```

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
proj	A redcapProject object as created by redcapProjectInfo.

Details

The function may only import a single file

Author(s)

Benjamin Nutter

Examples

```
## Not run:
> #*** Note: I cannot provide working examples without
> #*** compromising security.  Instead, I will try to
> #*** offer up sample code with the matching results
>
> #*** Create the connection object
> rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])
>
>
> #* Import a single file
> importFiles(rcon, "Image.jpg", record=1, field="file_upload", event="event_1_arm_1")
The file was successfully uploaded
>

## End(Not run)
```

---

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 = "", ..., proj = 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.
`proj`	A `redcapProject` object as created by `redcapProjectInfo`.
`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

http://stackoverflow.com/questions/12393004/parsing-back-to-messy-api-strcuture/12435389#12435389
https://github.com/etb/my-R-code/blob/master/R-pull-and-push-from-and-to-REDCap.R
See also the REDCap API documentation 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

See Also

validateImport

Examples

```
## Not run:
> #*** Note: I cannot provide working examples without
> #*** compromising security.  Instead, I will try to
> #*** offer up sample code with the matching results
>
>
> #*** Create the connection object
> rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])
>
>
> #*** Import a record for a new patient
> NewScan <- data.frame(patient_id = 1022,
+                       redcap_event_name = "entry_arm_1",
+                       bmi = 24.689,
+                       patient_characteristics_complete = 1)
>
> importRecords(rcon, NewScan)
REDCap Data Import Log: 2014-06-20 16:08:31
The following (if any) conditions were noted about the data.


1
> ## No conditions were noted, 1 record was uploaded
>
>
>
>
>
> #*** Import a record for a new patient with an erroneous BMI value
> NewScan <- data.frame(patient_id = 1022,
+                       redcap_event_name = "entry_arm_1",
+                       bmi = 244.689,
+                       patient_characteristics_complete = 4)
>
> importRecords(rcon, NewScan)
```

```
REDCap Data Import Log: 2014-06-20 16:08:33
The following (if any) conditions were noted about the data.


1022   entry_arm_1   244.689   Entry for 'bmi' is larger than
the acceptable maximum.  Please confirm.
1
> ## One condition was noted.  Notice that the BMI value was still
> ## uploaded to REDCap.

## End(Not run)
```

`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, level 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

Examples

```
## Not run:
> #*** Note: I cannot provide working examples without
> #*** compromising security.  Instead, I will try to
> #*** offer up sample code with the matching results
>
```

```
>
> #*** Create the connection object
> rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])
>
> #* Default appearance after export
> Prenatal <- exportRecords(rcon, fields=c("maternal_mrn", "consults"))
Warning message:
  In exportMetaData.redcapApiConnection(rcon) : NAs introduced by coercion
> head(Prenatal)
maternal_mrn   redcap_event_name consults___1
1 0d714b1efc778d8e738c7f8eb399d224 mfm_episode_1_arm_1    Unchecked
2 0ef1975c93365e2246038d317838816d mfm_episode_1_arm_1    Unchecked
3 a0d81f6f1e55de0770825f460e8e4894 mfm_episode_1_arm_1      Checked
4 a1a0a470c658d05e7df636607fc89bd4 mfm_episode_1_arm_1      Checked
5 a577d8066a12536adb97df9d66ad7d39 mfm_episode_1_arm_1      Checked
6 a5e9beb28c9883b8bd8961481064037e mfm_episode_1_arm_1    Unchecked
consults___2 consults___3 consults___4 consults___5 consults___6 consults___7
1    Unchecked      Checked    Unchecked    Unchecked      Checked    Unchecked
2      Checked      Checked    Unchecked    Unchecked    Unchecked    Unchecked
3    Unchecked      Checked    Unchecked    Unchecked    Unchecked    Unchecked
4    Unchecked    Unchecked    Unchecked    Unchecked    Unchecked    Unchecked
5    Unchecked      Checked    Unchecked    Unchecked    Unchecked    Unchecked
6    Unchecked    Unchecked    Unchecked    Unchecked    Unchecked    Unchecked
consults___8 consults___9 consults___10 consults___11 consults___12
1    Unchecked    Unchecked     Unchecked     Unchecked     Unchecked
2    Unchecked    Unchecked     Unchecked     Unchecked     Unchecked
3    Unchecked    Unchecked     Unchecked     Unchecked     Unchecked
4    Unchecked    Unchecked     Unchecked     Unchecked     Unchecked
5      Checked    Unchecked     Unchecked     Unchecked     Unchecked
6    Unchecked    Unchecked     Unchecked     Unchecked     Unchecked
consults___13 consults___14 consults___90
1     Unchecked     Unchecked     Unchecked
2     Unchecked     Unchecked       Checked
3     Unchecked     Unchecked     Unchecked
4     Unchecked     Unchecked     Unchecked
5     Unchecked     Unchecked     Unchecked
6     Unchecked     Unchecked       Checked
>
>
> #* Use the default settings to recode as No/Yes
> Prenatal2 <- recodeCheck(Prenatal)
> head(Prenatal2)
maternal_mrn   redcap_event_name consults___1
1 0d714b1efc778d8e738c7f8eb399d224 mfm_episode_1_arm_1           No
2 0ef1975c93365e2246038d317838816d mfm_episode_1_arm_1           No
3 a0d81f6f1e55de0770825f460e8e4894 mfm_episode_1_arm_1          Yes
4 a1a0a470c658d05e7df636607fc89bd4 mfm_episode_1_arm_1          Yes
5 a577d8066a12536adb97df9d66ad7d39 mfm_episode_1_arm_1          Yes
6 a5e9beb28c9883b8bd8961481064037e mfm_episode_1_arm_1           No
consults___2 consults___3 consults___4 consults___5 consults___6 consults___7
1           No          Yes           No           No          Yes           No
2          Yes          Yes           No           No           No           No
3           No          Yes           No           No           No           No
4           No           No           No           No           No           No
5           No          Yes           No           No           No           No
6           No           No           No           No           No           No
consults___8 consults___9 consults___10 consults___11 consults___12
```

```
1           No           No            No            No           No
2           No           No            No            No           No
3           No           No            No            No           No
4           No           No            No            No           No
5          Yes           No            No            No           No
6           No           No            No            No           No
consults___13 consults___14 consults___90
1            No            No            No
2            No            No           Yes
3            No            No            No
4            No            No            No
5            No            No            No
6            No            No           Yes
>
>
>
> #* Alter the defaults to recode as Received Consult/No Consult Necessary
> Prenatal3 <- recodeCheck(Prenatal,
+                          levels=c("Checked", "Unchecked"),
+                          labels=c("Received Consult", "No Consult Necessary"))
> head(Prenatal3)
maternal_mrn   redcap_event_name         consults___1
1 0d714b1efc778d8e738c7f8eb399d224 mfm_episode_1_arm_1 No Consult Necessary
2 0ef1975c93365e2246038d317838816d mfm_episode_1_arm_1 No Consult Necessary
3 a0d81f6f1e55de0770825f460e8e4894 mfm_episode_1_arm_1     Received Consult
4 a1a0a470c658d05e7df636607fc89bd4 mfm_episode_1_arm_1     Received Consult
5 a577d8066a12536adb97df9d66ad7d39 mfm_episode_1_arm_1     Received Consult
6 a5e9beb28c9883b8bd8961481064037e mfm_episode_1_arm_1 No Consult Necessary
consults___2         consults___3         consults___4
1 No Consult Necessary     Received Consult No Consult Necessary
2     Received Consult     Received Consult No Consult Necessary
3 No Consult Necessary     Received Consult No Consult Necessary
4 No Consult Necessary No Consult Necessary No Consult Necessary
5 No Consult Necessary     Received Consult No Consult Necessary
6 No Consult Necessary No Consult Necessary No Consult Necessary
consults___5         consults___6         consults___7
1 No Consult Necessary     Received Consult No Consult Necessary
2 No Consult Necessary No Consult Necessary No Consult Necessary
3 No Consult Necessary No Consult Necessary No Consult Necessary
4 No Consult Necessary No Consult Necessary No Consult Necessary
5 No Consult Necessary No Consult Necessary No Consult Necessary
6 No Consult Necessary No Consult Necessary No Consult Necessary
consults___8         consults___9        consults___10
1 No Consult Necessary No Consult Necessary No Consult Necessary
2 No Consult Necessary No Consult Necessary No Consult Necessary
3 No Consult Necessary No Consult Necessary No Consult Necessary
4 No Consult Necessary No Consult Necessary No Consult Necessary
5     Received Consult No Consult Necessary No Consult Necessary
6 No Consult Necessary No Consult Necessary No Consult Necessary
consults___11        consults___12        consults___13
1 No Consult Necessary No Consult Necessary No Consult Necessary
2 No Consult Necessary No Consult Necessary No Consult Necessary
3 No Consult Necessary No Consult Necessary No Consult Necessary
4 No Consult Necessary No Consult Necessary No Consult Necessary
5 No Consult Necessary No Consult Necessary No Consult Necessary
6 No Consult Necessary No Consult Necessary No Consult Necessary
consults___14        consults___90
```

```
1 No Consult Necessary No Consult Necessary
2 No Consult Necessary     Received Consult
3 No Consult Necessary No Consult Necessary
4 No Consult Necessary No Consult Necessary
5 No Consult Necessary No Consult Necessary
6 No Consult Necessary     Received Consult
>
>
>
> ## The order of the levels can be rearranged if desired
> levels(Prenatal2$consults___90)
[1] "No"  "Yes"
> levels(Prenatal3$consults___90)
[1] "Received Consult"     "No Consult Necessary"

## End(Not run)
```

---

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

Description

REDCap is a database development tool built on MySQL. Visit 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 Stevens at Vanderbilt University. Their work in developing the redcap package (http://github.com/vubiostat/redcap) 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

References

This functionality were originally developed by Jeffrey Horner in the redcap package. https://github.com/vubiostat/redcap

A tutorial on configuring the REDCap user rights for the API is found at https://github.com/nutterb/redcapAPI/wiki/Setting-the-User-Rights-to-Grant-API-Access

A tutorial on requesting and obtaining your API token is found at https://github.com/nutterb/redcapAPI/wiki/Finding-Your-REDCap-API-Token

A tutorial on finding your API url is found at https://github.com/nutterb/redcapAPI/wiki/Finding-your-REDCap-API-URL

A tutorial for finding and using alternate SSL certificates is found at https://github.com/nutterb/redcapAPI/wiki/Manually-Setting-an-SSL-Certificate-File

Examples

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

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

## 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

`redcapProjectInfo` *Export a Project's Supplemental Information*

Description

The meta data, users, arms, events, mappings, and REDCap version can be used to do some validation during data exports and imports. In order to reduce calls to the API, these tables can be stored in an object and referenced as needed.

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`	Character string. Can be used to manually set the REDCap version number for users running a version earlier than 6.0. This is used to validate calls that may only be available in certain versions.

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

Examples

```
## Not run:
> #*** Note: I cannot provide working examples without
> #*** compromising security.  Instead, I will try to
> #*** offer up sample code with the matching results
>
>
> #*** Create the connection object
> rcon <- redcapConnection(url=[YOUR_REDCAP_URL], token=[API_TOKEN])
>
> redcapProjectInfo(rcon)

## End(Not run)
```

`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.

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(field, meta_data, records, ids, logfile = "")
```

Arguments

`field`	Character(1) naming the variable to be validated.
`meta_data`	REDCap database meta data.
`records`	The data frame to be validated.
`ids`	Character vector giving the names of fields that uniquely identify a record. Usually the study id and `redcap_event_name`.
`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. Currently, these variables *are* 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'

Validating e-mail addresses http://www.regular-expressions.info/email.html

Index