Package 'REDCapDM'

January 24, 2026

Type Package

Title 'REDCap' Data Management

Version 1.0.1

Maintainer João Carmezim <jcarmezim@igtp.cat>

Description REDCap Data Management - 'REDCap' (Research Electronic Data CAPture; <https://projectredcap.org>) is a web application developed at Vanderbilt University, designed for creating and managing online surveys and databases and the REDCap API is an interface that allows external applications to connect to REDCap remotely, and is used to programmatically retrieve or modify project data or settings within REDCap, such as importing or exporting data. REDCapDM is an R package that allows users to manage data exported directly from REDCap or using an API connection. This package includes several functions designed for pre-processing data, generating reports of queries such as outliers or missing values, and following up on previously identified queries.

License MIT + file LICENSE

URL https://bruigtp.github.io/REDCapDM/,
https://doi.org/10.1186/s12874-024-02178-6

BugReports https://github.com/bruigtp/REDCapDM/issues

Encoding UTF-8

RoxygenNote 7.3.3

Imports dplyr, janitor, openxlsx, purrr, REDCapR, rlang, stringr, tibble, tidyr, tidyselect, labelled, utils, stringi, cli, forcats, lifecycle, magrittr

Suggests knitr, rmarkdown, kableExtra, testthat (>= 3.0.0), mockery

VignetteBuilder knitr

Depends R (>= 4.1)

LazyData true

Config/testthat/edition 3

Config/Needs/website rmarkdown

NeedsCompilation no

Author João Carmezim [aut, cre],
Pau Satorra [aut],
Judith Peñafiel [aut],
Esther García [aut],
Natàlia Pallarès [aut],
Cristian Tebé [aut]

Repository CRAN

Date/Publication 2026-01-24 16:20:02 UTC

Contents

`checkbox_names` *Change Checkboxes Names to Option Names*

Description

This function updates the names of checkboxes in the dataset and dictionary to reflect the names of their options.

Usage

```
checkbox_names(data, dic, labels, checkbox_labels = c("No", "Yes"))
```

Arguments

data	Dataset containing the REDCap data.
dic	Dataset containing the REDCap dictionary.
labels	Named character vector with the names of the variables in the data and their corresponding REDCap labels.
checkbox_labels	Character vector specifying the names for the two options of each checkbox variable. The default is `c('No', 'Yes')`.

---

check_proj *Handle Project Arguments*

Description

This helper function processes the `project` argument in the several other functions. It extracts data, dictionary, event form, and results from the project object while handling potential duplication and providing warnings when arguments are provided redundantly.

Usage

```
check_proj(project, data = NULL, dic = NULL, event_form = NULL)
```

Arguments

project	A list or object containing `data`, `dictionary`, and optionally `event_form` and `results`.
data	A data frame (optional) that may be overridden if provided in the `project` object.
dic	A data dictionary (optional) that may be overridden if provided in the `project` object.
event_form	An optional event-form object that may be overridden if provided in the `project`.

check_queries *Check for Changes Between Two Query Reports*

Description

[Stable]

Compare an older query report (`old`) with a newer one (`new`) and classify each query into one of four statuses:

- **Pending** — the same query is present in both reports (no change detected),
- **Solved** — the query was present in the old report but is absent from the new report,
- **New** — the query appears in the new report but was not present in the old report,
- **Miscorrected** — a special case where a query in the new report is marked as `New` but shares the same `Identifier` and `Description` as an existing record (suggesting a re-issued or modified query for the same identifier).

The function returns a detailed comparison table of queries with a `Modification` factor (one of the four statuses) and an HTML summary table showing counts per status.

Usage

```
check_queries(old, new, report_title = NULL, return_viewer = TRUE)
```

Arguments

`old`	Data frame containing the previous (older) query report. Must include `Identifier`, `Description` and `Query` columns (character or factor).
`new`	Data frame containing the newer query report. Must include `Identifier`, `Description` and `Query` columns (character or factor). If `new` contains a `Code` column, it will be removed at the start of processing.
`report_title`	Optional single string used as the caption for the HTML summary table. Defaults to `"Comparison report"` when not supplied or when `NA`.
`return_viewer`	Logical; if `TRUE` (default) an HTML table (knitr/kable + kableExtra) summarizing the counts per state is produced and returned in the `results` element of the returned list. If `FALSE`, no HTML viewer is produced (useful for non-interactive runs).

Details

Requirements:

- Both `old` and `new` must be data frames.
- Both data frames must contain at least the following character columns: `Identifier`, `Description`, and `Query`.

- A Code column is optional; if present it will be preserved and considered for sorting and output. If Code exists in new, it is removed at the beginning of the routine to avoid conflicts with re-assigned codes.

The function merges the two reports, constructs composite keys used for comparison, classifies each row into a modification state, detects and re-labels Miscorrected cases, reassigns a Code per Identifier to keep codes consistent, and returns a detailed dataset plus an optional HTML summary viewer.

Value

A list with two elements:

queries A data frame containing all queries present in either old or new. A factor column Modification indicates the state for each row (levels: Pending, Solved, Miscorrected, New). The function also reassigns Code values so codes are consistent per Identifier.

results If return_viewer = TRUE, an HTML knitr::kable (styled with kableExtra) summarising totals per state. If return_viewer = FALSE, this is NULL.

Notes and edge cases

- **Column types:** If Identifier, Description or Query are factors, they will be used in the comparison — it is recommended to convert them to character prior to calling check_queries() to avoid factor-level mismatches.
- **Sorting:** When Identifier values contain a dash (e.g. "100-20"), the function attempts to split into numeric center and id parts for logical ordering. Otherwise Identifier is coerced to numeric for ordering.
- **Miscorrected detection:** A Miscorrected label is assigned when more than one row shares the same Identifier + Description composite and a row is otherwise classified as New — this signals a likely re-issued or modified query for an existing identifier.

Examples

```
# Minimal reproducible example
old <- data.frame(
  Identifier = c("100-1", "100-2", "200-1"),
  Description = c("age check", "weight check", "lab miss"),
  Query = c("is.na(age)", "is.na(weight)", "missing lab result"),
  Code = c("100-1-1", "100-2-1", "200-1-1"),
  stringsAsFactors = FALSE
)

new <- data.frame(
  Identifier = c("100-1", "200-1", "300-1"),
  Description = c("age check", "lab miss", "new query"),
  Query = c("is.na(age)", "missing lab result (clarify)", "is.na(x)"),
  stringsAsFactors = FALSE
)

res <- check_queries(old = old, new = new, report_title = "My Query Comparison")
```

```
# detailed table
head(res$queries)
# HTML summary (if in an RMarkdown or interactive viewer)
res$results
```

`covican` *Subset of COVICAN's Database*

Description

A random sample of the COVICAN study. An international, multicentre cohort study of cancer patients with COVID-19 to describe the epidemiology, risk factors, and clinical outcomes of co-infections and superinfections in onco-haematological patients with COVID-19.

Usage

```
data(covican)
```

Format

A data frame with 342 rows and 56 columns

record_id: Identifier of each record. This information does not match the real data.

redcap_event_name: Auto-generated name of the events

redcap_data_access_group: Auto-generated name of each center. This information does not match the real data.

inc_1: Inclusion criteria of 'Patients older than 18 years' (0 = No ; 1 = Yes)

inc_2: Inclusion criteria of 'Cancer patients' (0 = No ; 1 = Yes)

inc_3: Inclusion criteria of 'Diagnosed of COVID-19' (0 = No ; 1 = Yes)

exc_1: Exclusion criteria of 'Solid tumour remission >1 year' (0 = No ; 1 = Yes)

screening_fail_crit: Indicator of non-compliance with inclusion and exclusion criteria (0 = compliance ; 1 = non-compliance)

d_birth: Date of birth (y-m-d). This date does not correspond to the original.

d_admission: Date of first visit (y-m-d). This date does not correspond to the original.

age: Age in years

dm: Indicator of diabetes (0 = No ; 1 = Yes)

type_dm: Type of diabetes (1 = No complications ; 2 = End-organ diabetes-related disease)

copd: Indicator of chronic obstructive pulmonary disease (0 = No ; 1 = Yes)

fio2: Fraction of inspired oxygen in percentage

available_analytics: Indicator of blood test available (0 = No ; 1 = Yes)

potassium: Potassium in mmol/L

resp_rate: Respiratory rate in bpm

leuk_lymph: Indicator of leukemia or lymphoma (0 = No ; 1 = Yes)

acute_leuk: Indicator of acute leukemia (0 = No ; 1 = Yes)

type_underlying_disease...: Checkbox with the type of underlying disease (0 = Haematological cancer ; 1 = Solid tumour)

underlying_disease_hemato...: Checkbox with the type of underlying disease (1 = Acute myeloid leukemia ; 2 = Myelodysplastic syndrome ; 3 = Chronic myeloid leukaemia ; 4 = Acute lymphoblastic leukaemia ; 5 = Hodgkin lymphoma ; 6 = Non Hodgkin lymphoma ; 7 = Multiple myeloma ; 8 = Myelofibrosis ; 9 = Aplastic anaemia ; 10 = Chronic lymphocytic leukaemia ; 11 = Amyloidosis ; 12 = Other)

urine_culture: Indicator of urine culture: (0 = Not done ; 1 = Done)

....factor: Labels of the different variables

Note

List with three data frames: the first one with the data, the second one with the dictionary (codebook) of the REDCap project and the last one with the instrument-event mappings of the REDCap project.

References

Gudiol, C., Durà-Miralles, X., Aguilar-Company, J., Hernández-Jiménez, P., Martínez-Cutillas, M., Fernandez-Avilés, F., Machado, M., Vázquez, L., Martín-Dávila, P., de Castro, N., Abdala, E., Sorli, L., Andermann, T. M., Márquez-Gómez, I., Morales, H., Gabilán, F., Ayaz, C. M., Kayaaslan, B., Aguilar-Guisado, M., Herrera, F. Royo-Cebrecos C, Peghin M, González-Rico C, Goikoetxea J, Salgueira S, Silva-Pinto A, Gutiérrez-Gutiérrez B, Cuellar S, Haidar G, Maluquer C, Marin M, Pallarès N, Carratalà J. (2021). Co-infections and superinfections complicating COVID-19 in cancer patients: A multicentre, international study. The Journal of infection, 83(3), 306–313. https://doi.org/10.1016/j.jinf.2021.07.014

`fill_data` *Fill Rows with Values from One Event*

Description

This function fills all rows in the dataset with the value of a particular variable in a specified event. It is an auxiliary function used in the `rd_rlogic` function.

Usage

```
fill_data(which_event, which_var, data)
```

Arguments

`which_event`	String specifying the name of the event.
`which_var`	String specifying the name of the variable.
`data`	Dataset containing the REDCap data.

rd_checkbox *Transform Checkbox Variables in a REDCap Project*

Description

[Experimental]

This function is used to process checkbox variables in a REDCap dataset. By default, it changes their default categories ("Unchecked" and "Checked") to new ones ("No" and "Yes). Optionally, the function can also evaluate the branching logic for checkbox fields and adjust the data and dictionary accordingly.

Usage

```
rd_checkbox(
  project = NULL,
  data = NULL,
  dic = NULL,
  event_form = NULL,
  checkbox_labels = c("No", "Yes"),
  checkbox_names = TRUE,
  na_logic = "missing"
)
```

Arguments

project	A list containing the REDCap data, dictionary, and event mapping (expected redcap_data() output). Overrides data, dic, and event_form.
data	A data.frame or tibble with the REDCap dataset.
dic	A data.frame with the REDCap dictionary.
event_form	Only applicable for longitudinal projects (presence of events). Event-to-form mapping for longitudinal projects.
checkbox_labels	Character vector of length 2 for labels of unchecked/checked values. Default: c("No", "Yes").
checkbox_names	Logical. If TRUE (default), checkbox columns are renamed using choice labels.
na_logic	Controls how missing values are set based on branching logic. Must be one of "none" (do nothing), "missing" (set to NA only when the logic evaluation is NA), or "eval" (set to NA when the logic evaluates to FALSE). Defaults to "missing".

Details

- Checkbox columns are expected in REDCap wide format (field___code).
- Branching logic evaluation requires event_form for longitudinal projects.
- Names are cleaned and truncated to 60 characters; uniqueness is enforced.
- Fields that cannot be evaluated are listed in results.

Value

A list with:

data Transformed dataset with checkbox fields as factors and optionally renamed.

dictionary Updated dictionary with checkbox fields expanded and optionally renamed.

event_form The `event_form` passed in (if applicable).

results Summary of transformations and any fields needing review.

Examples

```
# Basic usage with a project object
res <- rd_checkbox(covican)

# With custom labels
res <- rd_checkbox(data = covican$data,
                   dic = covican$dictionary,
                   checkbox_labels = c("Not present", "Present"))

# Keep original checkbox names
res <- rd_checkbox(covican, checkbox_names = FALSE)

# Longitudinal project with NA logic
res <- rd_checkbox(data = covican$data,
                   dic = covican$dictionary,
                   event_form = covican$event_form,,
                   na_logic = "eval")
```

---

rd_dates *Transform Dates and Datetimes in REDCap Data*

Description

[Experimental]

Converts date and datetime fields in a REDCap dataset to appropriate R classes.

Usage

```
rd_dates(project = NULL, data = NULL, dic = NULL, event_form = NULL)
```

Arguments

`project`	A list containing the REDCap data, dictionary, and event mapping (expected `redcap_data()` output). Overrides `data`, `dic`, and `event_form`.
`data`	A `data.frame` or `tibble` with the REDCap dataset.
`dic`	A `data.frame` with the REDCap dictionary.
`event_form`	Only applicable for longitudinal projects (presence of events). Event-to-form mapping for longitudinal projects.

Details

The function performs the following tasks:

- Detects date and datetime fields from the REDCap dictionary (`date_*` and `datetime_*` validation types).
- Converts date fields to `Date` class.
- Converts datetime fields to `POSIXct` class, treating empty strings as `NA`.

Value

A list with the following elements:

data The transformed dataset with date and datetime fields formatted as `Date` and `POSIXct`.

dictionary The original REDCap dictionary passed to the function.

event_form The original event-form mapping (if applicable).

results A summary of the transformations performed.

Examples

```
result <- rd_dates(covican)
transformed_data <- result$data
```

---

rd_delete_vars *Delete Variables from REDCap Dataset and Dictionary*

Description

[Experimental]

Deletes selected variables from a REDCap dataset and its dictionary, keeping them consistent and preserving variable labels.

Usage

```
rd_delete_vars(
  project = NULL,
  data = NULL,
  dic = NULL,
  event_form = NULL,
  vars = NULL,
  pattern = NULL
)
```

Arguments

project	A list containing the REDCap data, dictionary, and event mapping (expected redcap_data() output). Overrides data, dic, and event_form.
data	A data.frame or tibble with the REDCap dataset.
dic	A data.frame with the REDCap dictionary.
event_form	Only applicable for longitudinal projects (presence of events). Event-to-form mapping for longitudinal projects.
vars	Optional. A character vector of variable names to remove from both the dataset and dictionary.
pattern	Optional. A character vector of regular expression patterns. Variables matching these patterns will be removed from the dataset and dictionary.

Details

- Ensure that at least one of vars or pattern is specified.
- Removes specified variables and their factor versions (e.g., variable.factor) from the dataset.
- Removes matching variables from the dictionary.
- Warns about factor versions of variables matching patterns, recommending use of rd_factor() if necessary.

Value

A list containing:

data The updated dataset with specified variables removed.

dictionary The updated REDCap dictionary.

event_form The original event-form mapping (if applicable).

results A summary message describing the variable removal operation.

Examples

```
# Delete specific variables by name
result <- rd_delete_vars(
  project = covican,
  vars = c("potassium", "leuk_lymph")
)

# Delete variables matching patterns
result <- rd_delete_vars(
  data = covican$data,
  dic = covican$dictionary,
  pattern = c("_complete$", "_other$")
)
```

`rd_dictionary` *Transform the data dictionary and handle branching logic*

Description

[Experimental]

Updates a REDCap data dictionary by converting branching logic and calculation expressions into valid R expressions. This ensures that conditional display rules and calculated fields in the dictionary can be programmatically evaluated with the dataset. Any variables that cannot be converted are reported in the results.

Usage

```
rd_dictionary(project = NULL, data = NULL, dic = NULL, event_form = NULL)
```

Arguments

`project`	A list containing the REDCap data, dictionary, and event mapping (expected `redcap_data()` output). Overrides `data`, `dic`, and `event_form`.
`data`	A `data.frame` or `tibble` with the REDCap dataset.
`dic`	A `data.frame` with the REDCap dictionary.
`event_form`	Only applicable for longitudinal projects (presence of events). Event-to-form mapping for longitudinal projects.

Details

The function performs the following tasks:

- Evaluates and transforms branching logic expressions into valid R expressions using `rd_rlogic`.
- Evaluates and converts calculation expressions for fields of type `calc`.
- Generates a results summary table listing any variables that could not be converted.

Value

A list with the following elements:

data The original dataset passed to the function.

dictionary The updated data dictionary with modified branching logic and calculations.

event_form The original event-form mapping (if applicable).

results A summary of the transformations, including any variables with unconverted branching logic.

Examples

```
## Not run:
result <- rd_dictionary(covican)
print(result$results)
updated_dic <- result$dictionary

## End(Not run)
```

`rd_event` *Identify Missing Events in REDCap Data*

Description

[Stable]

Helps identify records in a REDCap longitudinal project that are missing one or more specified events. Because REDCap typically omits empty events from exports, an event that contains no data for a record will not appear. This function finds those absent events and returns a per-record query table and a summarized HTML report.

Usage

```
rd_event(
  project = NULL,
  data = NULL,
  dic = NULL,
  event_form = NULL,
  event,
  filter = NA,
  query_name = NA,
  addTo = NA,
  report_title = NA,
  report_zeros = FALSE,
  link = list()
)
```

Arguments

`project`	A list containing the REDCap data, dictionary, and event mapping (expected `redcap_data()` output). Overrides `data`, `dic`, and `event_form`.
`data`	A `data.frame` or `tibble` with the REDCap dataset.
`dic`	A `data.frame` with the REDCap dictionary.
`event_form`	Only applicable for longitudinal projects (presence of events). Event-to-form mapping for longitudinal projects.
`event`	Character vector with one or more REDCap event names to check for missing records.

filter	Optional. A single character string containing a filter expression to subset the dataset before checking for missing events. Example: "age >= 18".
query_name	Optional character vector describing each query. Defaults to a standard format: The event (event_name) is missing.
addTo	Optional data frame from a previous query report to which the new results can be appended.
report_title	Optional string specifying the title of the final report. Defaults to "Report of queries".
report_zeros	Logical, include variables with zero queries in the report. Default is FALSE.
link	Optional list containing project information (domain, redcap_version, proj_id, event_id) to generate clickable links for each query.

Value

A named list with two elements:

queries A data frame listing records missing the specified events. Columns: Identifier, DAG, Event, Instrument, Field, Repetition, Description, Query, Code, and optionally Link. If no queries are found this will be an empty data frame with the expected columns.

results An HTML table (knitr::kable styled with kableExtra) summarising the number of missing records per event. Returned as knitr::kable (HTML).

Examples

```
# Minimal reproducible example
data0 <- data.frame(
  record_id = c("100-1", "100-2", "200-1"),
  redcap_event_name = c("baseline_arm_1", "baseline_arm_1", "follow_up_arm_1"),
  redcap_event_name.factor = factor(c("Baseline", "Baseline", "Follow-up")),
  stringsAsFactors = FALSE
)

# Suppose we want to check that every record has the follow-up event
res <- rd_event(
  data = data0,
  dic = data.frame(),       # placeholder dictionary
  event = "follow_up_arm_1",
  report_zeros = TRUE
)

# Records missing the event:
res$queries

# HTML summary (in RMarkdown or Viewer)
res$results
```

rd_export *Export Queries to an Excel File*

Description

[Experimental]

Export a query dataset (e.g., from `rd_query` or `rd_event`) to an `.xlsx` file. The function can optionally convert a column of URLs into Excel hyperlinks and apply password protection to the worksheet.

Usage

```
rd_export(
  project = NULL,
  queries = NULL,
  column = NULL,
  sheet_name = NULL,
  path = NULL,
  password = NULL
)
```

Arguments

`project`	A list containing the dataframe of queries and results (expected `rd_query` or `rd_event` output). Overrides `queries`.
`queries`	A data frame of identified queries.
`column`	Name of the column containing URLs to convert into hyperlinks. If `NULL`, hyperlinks are added only if a `Link` column exists.
`sheet_name`	Name of the Excel sheet in the resulting `.xlsx` file. Default: `"Sheet1"`.
`path`	File path for saving the `.xlsx` file. If `NULL`, the file is saved as `"example.xlsx"` in the working directory.
`password`	Optional password to protect the worksheet from edits.

Value

An `.xlsx` file written to the specified path.

Examples

```
## Not run:
rd_export(
  queries = my_queries,
  column = "Link",
  sheet_name = "My Queries",
  path = "queries.xlsx"
)
```

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

`rd_factor` *Convert Variables to Factors in a REDCap Dataset*

Description

[Experimental]

Converts variables in a REDCap dataset with associated `.factor` columns into actual factor variables, while allowing the exclusion of specific variables. Ensures consistency with the dataset and preserves variable labels.

Usage

```
rd_factor(
  project = NULL,
  data = NULL,
  dic = NULL,
  event_form = NULL,
  exclude = NULL
)
```

Arguments

`project`	A list containing the REDCap data, dictionary, and event mapping (expected `redcap_data()` output). Overrides `data`, `dic`, and `event_form`.
`data`	A `data.frame` or `tibble` with the REDCap dataset.
`dic`	A `data.frame` with the REDCap dictionary.
`event_form`	Only applicable for longitudinal projects (presence of events). Event-to-form mapping for longitudinal projects.
`exclude`	Optional character vector of variable names (use original names **without** the `.factor` suffix) to exclude from conversion.

Details

The function looks for columns ending in `.factor` and replaces the original variable values with those `.factor` values (converted to factors). It preserves variable labels. The `exclude` argument must contain base variable names (no `.factor` suffix); if any `.factor` names are passed to `exclude` the function will throw an informative error. The columns `redcap_event_name`, `redcap_repeat_instrument` and `redcap_data_access_group` (and their `.factor` counterparts) are handled specially to avoid altering event or access-group data.

Value

A list with the following elements:

data The transformed dataset with `.factor` columns applied as factors.

dictionary The dictionary used (unchanged).

event_form The event-form mapping used (if applicable).

results A brief text summary of the transformation.

Examples

```
## Not run:
result <- rd_factor(covican)
result <- rd_factor(covican, exclude = c("available_analytics", "urine_culture"))
transformed_data <- result$data

## End(Not run)
```

`rd_insert_na` *Insert Missing Values Using a Filter*

Description

[Stable]

Sets selected variables to `NA` when a filter condition is satisfied. Useful for managing checkboxes or other fields without explicit gatekeeper questions.

Usage

```
rd_insert_na(
  project = NULL,
  data = NULL,
  dic = NULL,
  event_form = NULL,
  vars,
  filter
)
```

Arguments

`project`	A list containing the REDCap data, dictionary, and event mapping (expected `redcap_data()` output). Overrides `data`, `dic`, and `event_form`.
`data`	A `data.frame` or `tibble` with the REDCap dataset.
`dic`	A `data.frame` with the REDCap dictionary.
`event_form`	Only applicable for longitudinal projects (presence of events). Event-to-form mapping for longitudinal projects.

vars	Character vector of variable names to set to NA.
filter	A single logical expression (as string). Rows where the filter evaluates to TRUE will have the corresponding vars set to NA.

Details

- Each variable is only updated in rows/events where both the variable and filter are present.
- For longitudinal projects, event_form must be provided for proper event-level filtering.
- Only one filter expression is allowed.
- Variables and filter columns must exist in both data and dictionary.

Value

A list with:

data The dataset with NA inserted where the filter applies.

dictionary The unchanged dictionary.

event_form The event_form passed in (if applicable).

results Summary message of the changes applied.

Examples

```
# Set 'potassium' to NA where age < 65
## Not run:
data <- rd_insert_na(
  data = covican$data,
  dic = covican$dictionary,
  vars = "potassium",
  filter = "age < 65"
)
table(data$potassium)

## End(Not run)
```

---

rd_query *Identification of Queries*

Description

[Stable]

Detects and summarizes queries in a REDCap dataset based on specified expressions, filters, or a defined branching logic. Useful for identifying missing values, out-of-range values, or values that do not meet predefined criteria.

Usage

```
rd_query(
  project = NULL,
  variables = NA,
  expression = NA,
  negate = FALSE,
  event = NA,
  filter = NA,
  addTo = NA,
  variables_names = NA,
  query_name = NA,
  instrument = NA,
  report_title = NA,
  report_zeros = FALSE,
  by_dag = FALSE,
  link = list(),
  data = NULL,
  dic = NULL,
  event_form = NULL
)
```

Arguments

project	A list containing the REDCap data, dictionary, and event mapping (expected redcap_data() output). Overrides data, dic, and event_form.
variables	Character vector of variable names to check for queries.
expression	Character vector of R expressions to evaluate for each variable.
negate	Logical, if TRUE, identifies values that **do not** meet the condition. Default is FALSE.
event	Required for longitudinal projects to avoid overestimation. REDCap event(s) to analyze.
filter	Optional string of filters to apply to the dataset, such as the branching logic of a variable.
addTo	Optional data frame from a previous query report to which the new results can be appended.
variables_names	Optional character vector of descriptions for each variable. Defaults to the variables labels in the dictionary.
query_name	Optional character vector describing each query. Defaults to a standard format: The value is [value] and it should not be [expression].
instrument	Optional REDCap instrument(s) for each variable. Defaults to the instrument reported in the dictionary.
report_title	Optional string specifying the title of the final report. Defaults to "Report of queries".
report_zeros	Logical, include variables with zero queries in the report. Default is FALSE.

by_dag	Logical, split results by Data Access Group (DAG). Default is FALSE.
link	Optional list containing project information (domain, redcap_version, proj_id, event_id) to generate clickable links for each query.
data	A data.frame or tibble with the REDCap dataset.
dic	A data.frame with the REDCap dictionary.
event_form	Only applicable for longitudinal projects (presence of events). Event-to-form mapping for longitudinal projects.

Details

The function performs the following steps:

- Applies user-specified expressions to the selected variables to detect queries.
- Optionally negates the expressions to find values that **do not** satisfy the condition.
- Handles REDCap branching logic, converting it into R-compatible expressions for evaluation.
- Applies additional user-specified filters before identifying queries.
- Generates structured query results with metadata including:
  - Identifier (record_id)
  - DAG (if present)
  - Event and Instrument
  - Field, Repetition, Description, Query statement
  - Optional link to REDCap entry
- Optionally combines results with previous query outputs using addTo.
- Produces a summarized report, optionally including variables with zero queries.
- Provides warnings for variables with branching logic that could not be automatically evaluated.

Value

A list containing:

queries A data frame or a list of data frames (if by_dag = TRUE) with detailed query information for each record.

results A formatted report (HTML table) summarizing total queries per variable, event, and DAG if applicable.

Examples

```
## Not run:
# Identify missing values for multiple variables
result <- rd_query(covican,
  variables = c("copd", "age"),
  expression = c("is.na(x)", "x %in% NA"),
  event = "baseline_visit_arm_1"
)
result$results
```

```
# Identify values exceeding a threshold
result <- rd_query(covican,
  variables = "age",
  expression = "x > 20",
  event = "baseline_visit_arm_1"
)

# Apply a filter to select subset of data
result <- rd_query(covican,
  variables = "potassium",
  expression = "is.na(x)",
  event = "baseline_visit_arm_1",
  filter = "available_analytics == '1'"
)

## End(Not run)
```

`rd_recalculate` *Recalculate and Verify Calculated Fields in REDCap Data*

Description

[Experimental]

Recalculates fields marked as calculated in the REDCap dictionary, compares them with the original values, and reports discrepancies. If any differences are found, new recalculated fields are added to the dataset and dictionary with `_recalc` appended to the names.

Usage

```
rd_recalculate(
  project = NULL,
  data = NULL,
  dic = NULL,
  event_form = NULL,
  exclude = NULL
)
```

Arguments

`project`	A list containing the REDCap data, dictionary, and event mapping (expected `redcap_data()` output). Overrides `data`, `dic`, and `event_form`.
`data`	A `data.frame` or `tibble` with the REDCap dataset.
`dic`	A `data.frame` with the REDCap dictionary.
`event_form`	Only applicable for longitudinal projects (presence of events). Event-to-form mapping for longitudinal projects.
`exclude`	Optional. Character vector of field names to exclude from recalculation.

Details

- Fields of type `calc` in the dictionary are recalculated.
- Recalculated values are compared with the original values.
- If differences exist, new fields `[field_name]_recalc` are added to the dataset and dictionary.
- Works for single-event projects; for longitudinal projects, `event_form` must be provided.
- Fields with incomplete branching logic or smart variables may fail to recalculate.

Value

A list with:

data The dataset with new `_recalc` fields for any differing calculated fields.

dictionary Updated dictionary including the new `_recalc` fields.

event_form The `event_form` passed in (if applicable).

results Summary report of the recalculation process, including tables of discrepancies.

Examples

```
# Recalculate all calculated fields
## Not run:
results <- rd_recalculate(
  data = covican$data,
  dic = covican$dictionary,
  event_form = covican$event_form
)

## End(Not run)

# Recalculate but exclude some variables
## Not run:
results <- rd_recalculate(
  project = covican,
  exclude = c("age", "screening_fail_crit")
)

## End(Not run)
```

---

rd_rlogic *Translate REDCap Logic to R Logic*

Description

[Stable]

Converts a REDCap logic expression into R-compatible logic. Processes one logic expression (`logic`) for one target variable (`var`) at a time. Supports common REDCap operators (`and`, `or`, `=`, `<`, `>`, etc.) and handles event-specific logic in longitudinal projects. Logic involving smart variables or repeated instruments may require manual review.

Usage

```
rd_rlogic(
  project = NULL,
  data = NULL,
  dic = NULL,
  event_form = NULL,
  logic,
  var
)
```

Arguments

project	A list containing the REDCap data, dictionary, and event mapping (expected redcap_data() output). Overrides data, dic, and event_form.
data	A data.frame or tibble with the REDCap dataset.
dic	A data.frame with the REDCap dictionary.
event_form	Only applicable for longitudinal projects (presence of events). Event-to-form mapping for longitudinal projects.
logic	A single REDCap logic string (e.g., "if([exc_1]='1' or [inc_1]='0', 1, 0)").
var	A single string specifying the target variable the logic applies to.

Details

- Translates REDCap operators and functions into R equivalents:
  - = → ==, <> → !=, and → &, or → |.
  - Converts functions like if(), rounddown(), datediff(), sum() to R equivalents.
- Handles date transformations and empty strings ('') → NA.
- Adjusts logic for longitudinal data using event_form if provided.
- Evaluates the translated R logic against the dataset and returns the results.
- Logic with repeated instruments, smart variables, or multiple events per variable may require manual inspection.

Value

A list with:

rlogic The translated R-compatible logic as a string.

eval The evaluation of the translated logic on the provided dataset, filtered by event if applicable.

Examples

```
# Translate a single REDCap logic expression for one variable
covican |>
  rd_rlogic(
    logic = "if([exc_1]='1' or [inc_1]='0' or [inc_2]='0' or [inc_3]='0', 1, 0)",
```

```
  var = "screening_fail_crit"
)
```

`rd_split` *Split a REDCap dataset by form or event*

Description

[Experimental]

Splits a REDCap dataset into separate datasets by **form** or **event** using the data dictionary. Supports both longitudinal and non-longitudinal projects and can return wide or long formats for repeated measures.

Usage

```
rd_split(
  project = NULL,
  data = NULL,
  dic = NULL,
  event_form = NULL,
  which = NULL,
  by = "form",
  wide = FALSE
)
```

Arguments

`project`	A list containing the REDCap data, dictionary, and event mapping (expected `redcap_data()` output). Overrides `data`, `dic`, and `event_form`.
`data`	A `data.frame` or `tibble` with the REDCap dataset.
`dic`	A `data.frame` with the REDCap dictionary.
`event_form`	Only applicable for longitudinal projects (presence of events). Event-to-form mapping for longitudinal projects.
`which`	Optional. A single form or event to extract. If not provided, all forms or events are returned.
`by`	Character. Criteria to split the dataset: `"form"` (default) or `"event"`.
`wide`	Logical. If `TRUE` (for form-based splits), repeated instances are returned in wide format. Defaults to `FALSE`.

Details

- Handles checkbox variables and REDCap default variables (`_complete`, `_timestamp`) appropriately.
- For form-based splits in longitudinal projects, uses `event_form` to map variables to events.
- Wide format expands repeated instances into multiple columns per record.
- Filtering by `which` allows extracting a single form or event.
- Projects with repeated instruments are handled by filtering on the `redcap_repeat_instrument` variable.

Value

Depending on `which` and `wide`:

data A `data.frame` or a list of `data.frames` representing the split datasets.

dictionary The original REDCap dictionary.

event_form The original event-form mapping (if applicable).

results A summary message of the splitting operation.

Examples

```
# Split by form and return wide format
result <- covican |>
  rd_split(by = "form", wide = TRUE)

print(result)

# Split by event (long format)
result <- covican |>
  rd_split(by = "event")

print(result)
```

`rd_transform` *Transformation of the Raw Data*

Description

[Stable]

Transforms the raw REDCap data read by the `redcap_data` function. The function runs in one-step pipeline all functions dedicated to processing the data and returns the transformed data and dictionary, along with a summary of each step done.

Usage

```
rd_transform(
  project = NULL,
  data = NULL,
  dic = NULL,
  event_form = NULL,
  checkbox_labels = c("No", "Yes"),
  na_logic = "none",
  exclude_recalc = NULL,
  exclude_factor = NULL,
  delete_vars = NULL,
  delete_pattern = NULL,
  final_format = "raw",
  which_event = NULL,
  which_form = NULL,
  wide = NULL
)
```

Arguments

project	A list containing the REDCap data, dictionary, and event mapping (expected redcap_data() output). Overrides data, dic, and event_form.
data	A data.frame or tibble with the REDCap dataset.
dic	A data.frame with the REDCap dictionary.
event_form	Only applicable for longitudinal projects (presence of events). Event-to-form mapping for longitudinal projects.
checkbox_labels	Character vector of length 2 for labels of unchecked/checked values. Default: c("No", "Yes").
na_logic	Controls how missing values are set based on the branching logic of a checkbox. Must be one of "none" (do nothing), "missing" (set to NA only when the logic evaluation is NA), or "eval" (set to NA when the logic evaluates to FALSE). Defaults to "none".
exclude_recalc	Optional. Character vector of field names to exclude from recalculation.
exclude_factor	Optional character vector of variable names (use original names **without** the .factor suffix) to exclude from conversion.
delete_vars	Optional. A character vector of variable names to remove from both the dataset and dictionary.
delete_pattern	Optional. A character vector of regular expression patterns. Variables matching these patterns will be removed from the dataset and dictionary.
final_format	Character string indicating the final format of the data. Options are raw, by_event or by_form. raw (default) returns the transformed data in its original structure, by_event returns it as a nested data frame by event, and by_form returns it as a nested data frame by form.
which_event	Character. If final_format = "by_event", return only this event.

`which_form`	Character. If `final_format = "by_form"`, return only this form.
`wide`	Logical. If `TRUE` (for form-based splits), repeated instances are returned in wide format. Defaults to `FALSE`.

Value

A list with elements:

data Transformed data (data.frame or nested list when split).

dictionary Updated dictionary data.frame.

event_form Event–form mapping (if applicable).

results Character summary of transformation steps performed.

Examples

```
# Minimal usage (project object or data + dictionary)
trans <- rd_transform(covican)

# Custom checkbox labels
trans <- rd_transform(covican,
                      checkbox_labels = c("Not present", "Present"))

# Return only a single form (wide)
trans <- rd_transform(covican,
                      final_format = "by_form",
                      which_form = "laboratory_findings",
                      wide = TRUE)
```

`recalculate` *Recalculate REDCap Calculated Fields*

Description

This function recalculates each calculated field if the logic can be transcribed to R. Note that calculated fields containing smart-variables or variables from other events cannot be transcribed.

The function returns the dataset and dictionary with the recalculated variables appended (named as the original field plus `_recalc`), along with a summary table of the recalculation results.

Usage

```
recalculate(data, dic, event_form = NULL, exclude_recalc = NULL)
```

Arguments

data	Data frame containing data from REDCap.
dic	Data frame containing the dictionary read from REDCap.
event_form	Data frame containing the correspondence of each event with each form.
exclude_recalc	Character vector with the names of the variables that should not be recalculated. Useful for projects with time-consuming recalculations for certain calculated fields.

---

redcap_data *Read REDCap data*

Description

[Stable]

Import REDCap data into R either from REDCap's exported R file or directly via the REDCap API. The function returns a list with the dataset, the project dictionary (metadata) and, for longitudinal projects, the instrument–event mapping (`event_form`) when available.

Usage

```
redcap_data(
  data_path = NA,
  dic_path = NA,
  event_path = NA,
  uri = NA,
  token = NA,
  sep = ",",
  filter_field = NULL,
  survey_fields = FALSE
)
```

Arguments

data_path	Path to exported R file (use with `dic_path`).
dic_path	Path to the dictionary file (CSV or XLSX; use with `data_path`)..
event_path	Path to the event-form mapping file (CSV or XLSX) for longitudinal projects (downloadable via the `Designate Instruments for My Events` tab within the `Project Setup` section of REDCap).
uri	REDCap API base URI (use with `token`).
token	REDCap API token (use with `uri`).
sep	Character string specifying the field separator for the exported dictionary CSV file. Defaul `","`.
filter_field	Optional character vector of field names to request from the API.
survey_fields	Logical; include survey-related fields when pulling via API. Default `FALSE`.

Details

Two import modes are supported:

- **Exported files** — use `data_path` (REDCap R export) and `dic_path` (dictionary CSV/XLSX).
- **API** — use `uri` and `token` to pull data and metadata directly from REDCap.

If the project is longitudinal, provide `event_path` (instrument–event mapping) or the function will attempt to fetch it from the API when using API mode.

Steps for using exported data in REDCap:

1. Use the REDCap *Export Data* function and choose *R Statistical Software* format.
2. REDCap generates:
   - A CSV file with observations.
   - An R script to format variables for import.
3. Ensure the exported files, dictionary, and event mapping (if any) are in the same directory.

Value

A list with:

- `data`: Imported dataset.
- `dictionary`: Variable dictionary (project metadata).
- `event_form`: Event-form mapping for longitudinal projects (if applicable).

Note

To use other package functions effectively, include the `dic_path` argument to load the project dictionary.

- Use either exported-files mode (`data_path` + `dic_path`) **or** API mode (`uri` + `token`) — not both.
- For exported files, REDCap's R export is required for `data_path`. Dictionary and event files must be CSV or XLSX.

Examples

```
## Not run:
# From exported files
out <- redcap_data(
  data_path = "project_export.r",
  dic_path  = "project_dictionary.csv",
  event_path = "instrument_event_map.csv"
)

# From API
out_api <- redcap_data(
  uri   = "https://redcap.example.org/api/",
  token = "REPLACE_WITH_TOKEN"
)
```

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

---

round *Round Numbers to a Specified Number of Digits —-*

Description

This function rounds numeric values to the specified number of decimal digits, mimicking the behavior of the base R `round()` function but implemented manually.

Usage

```
round(x, digits)
```

Arguments

`x`	A numeric vector to be rounded.
`digits`	Integer indicating the number of decimal places to round to.

Value

A numeric vector rounded to the specified number of digits.

Examples

```
round(3.14159, 2)
round(c(-2.718, 3.14159), 1)
```

---

split_event *Creation of a Data Frame with Variables from All Forms of a Specified Event*

Description

This function generates a nested dataset filtered by each event, containing only the variables associated with each event. It uses the provided data, dictionary, and event-form mapping. You can choose to return data for a specific event.

Usage

```
split_event(data, dic, event_form, which = NULL)
```

Arguments

data	Data frame containing data from REDCap.
dic	Data frame containing the dictionary read from REDCap.
event_form	Data frame containing the correspondence of each event with each form.
which	Character string specifying an event if only data for that event is desired.

---

split_form *Creation of a Data Frame with Variables from a Specified Form*

Description

This function generates a nested dataset containing only the variables associated with each form, using the provided data, dictionary, and event-form mapping. You can choose to return data for a specific form.

Usage

```
split_form(data, dic, event_form = NULL, which = NULL, wide = FALSE)
```

Arguments

data	Data frame containing data from REDCap.
dic	Data frame containing the dictionary read from REDCap.
event_form	Data frame containing the correspondence of each event with each form.
which	Character string specifying a form if only data for that form is desired.
wide	Logical indicating if the dataset should be returned in a wide format (TRUE) or long format (FALSE).

---

to_factor *Convert Variables to Factors*

Description

This function converts all variables in the dataset to factors, except those specified in the exclude parameter.

Usage

```
to_factor(data, dic, exclude = NULL)
```

Arguments

data	Data frame containing the REDCap data.
dic	Data frame containing the REDCap dictionary.
exclude	Character vector specifying the names of variables that should not be converted to factors. If NULL, all variables will be converted.

`transform_checkboxes` *Transformation of Checkboxes with Branching Logic*

Description

This function inspects all the checkboxes in the study to determine if they have a branching logic. If a branching logic is present and its result is missing, the function will input a missing value into the checkbox. If `checkbox_na` is `TRUE`, the function will additionally input a missing value when the branching logic isn't satisfied, not just when it is missing. If a branching logic cannot be found or the logic cannot be transcribed due to the presence of smart variables, the variable is added to a list of reviewable variables that will be printed.

The function returns the dataset with the transformed checkboxes and a table summarizing the results.

Usage

```
transform_checkboxes(data, dic, event_form = NULL, checkbox_na = FALSE)
```

Arguments

`data`	Data frame containing data from REDCap.
`dic`	Data frame containing the dictionary read from REDCap.
`event_form`	Data frame containing the correspondence of each event with each form.
`checkbox_na`	Logical indicating if values of checkboxes with branching logic should be set to missing only when the branching logic is missing (`FALSE`), or also when the branching logic is not satisfied (`TRUE`). The default is `FALSE`.

Index