Package 'redquack'

October 9, 2025

Title Transfer 'REDCap' Data to Database

Version 0.3.0

Description Transfer 'REDCap' (Research Electronic Data Capture) data to a database, specifically optimized for 'DuckDB'.
Processes data in chunks to handle large datasets without exceeding available memory.
Features include data labeling, coded value conversion, and hearing a ``quack'' sound on success.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports audio, cli, DBI, dplyr, dbplyr, duckdb, httr2, labelled, readr, rlang

Suggests arrow, keyring, pak, RSQLite, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 4.1.0)

NeedsCompilation no

Author Dylan Pieper [aut, cre]

Maintainer Dylan Pieper <dylanpieper@gmail.com>

Repository CRAN

Date/Publication 2025-10-09 16:10:02 UTC

Contents

`close_duckdb` *Close DuckDB Connection*

Description

Closes a DuckDB connection.

Usage

```
close_duckdb(conn)
```

Arguments

`conn`	A DuckDB connection object.

Value

Invisible NULL.

Examples

```
## Not run:
conn <- use_duckdb()
# Use the connection...
close_duckdb(conn)

## End(Not run)
```

`collect_labeled` *Collect Labeled Data from Database Table*

Description

Collect data from a database table reference (tbl) and apply column and/or value labels from REDCap metadata using the labelled package. This function works in the tidy style with dplyr.

Usage

```
collect_labeled(
  data,
  cols = TRUE,
  vals = TRUE,
  convert = TRUE,
  metadata_table_name = "metadata"
)
```

Arguments

data	A tbl_sql object (database table reference) to apply labels to. The connection and table name are automatically extracted.
cols	Logical indicating whether to apply column (variable) labels. Default is TRUE.
vals	Logical indicating whether to apply value labels to coded variables. Default is TRUE.
convert	Logical indicating whether to convert coded values to their text labels (e.g., 0/1 becomes "No"/"Yes"). Default is TRUE.
metadata_table_name	Character string specifying the metadata table name. Default is "metadata".

Value

A data frame with labels applied according to the cols and vals parameters. If convert = TRUE, coded values are converted to text.

Examples

```
## Not run:
library(redquack)
duckdb <- DBI::dbConnect(duckdb::duckdb(), "redcap.duckdb")

# Apply both column and value labels (default)
labeled_data <- tbl_redcap(duckdb, "data") |> collect_labeled()

# Apply only column labels
col_labeled_data <- tbl_redcap(duckdb, "data") |> collect_labeled(vals = FALSE)

# Apply only value labels
val_labeled_data <- tbl_redcap(duckdb, "data") |> collect_labeled(cols = FALSE)

# Apply labels and convert values to text
labeled_data <- tbl_redcap(duckdb, "data") |> collect_labeled(convert = TRUE)

# Explicit metadata table name (useful after complex filtering)
labeled_data <- tbl(duckdb, "data") |>
  dplyr::filter(name_last == "Nutmouse") |>
  collect_labeled(metadata_table_name = "metadata")
```

```
DBI::dbDisconnect(duckdb)

## End(Not run)
```

`collect_list` *Collect a Database Table into to List of REDCap Instruments*

Description

Takes a database table reference (tbl) and collects it into a list of instruments with column and value labels and optional coded value conversion. This function works in the tidy style with dplyr and separates the data by REDCap instruments. Use `collect_labeled_list` as an alias for the same functionality.

Usage

```
collect_list(
  data,
  cols = FALSE,
  vals = FALSE,
  convert = FALSE,
  metadata_table_name = "metadata"
)

collect_labeled_list(
  data,
  cols = TRUE,
  vals = TRUE,
  convert = TRUE,
  metadata_table_name = "metadata"
)
```

Arguments

`data`	A tbl object referencing a database table (created with `tbl(conn, "data")`).
`cols`	Logical indicating whether to apply column (variable) labels. Default is FALSE.
`vals`	Logical indicating whether to apply value labels to coded variables. Default is FALSE.
`convert`	Logical indicating whether to convert labeled values to their text labels (e.g., 0/1 becomes "No"/"Yes"). Default is FALSE.
`metadata_table_name`	Character string specifying the metadata table name. Default is "metadata".

Value

Returns a named list of data frames, one per instrument. If only one instrument is present, returns the single data frame instead of a list.

See Also

redcap_to_db for transferring data to a database collect_labeled for collecting labeled data as a single data frame

Examples

```
## Not run:
library(redquack)
library(dplyr)

conn <- use_duckdb()

result <- redcap_to_db(
  conn,
  url = "https://bbmc.ouhsc.edu/redcap/api/",
  token = "9A81268476645C4E5F03428B8AC3AA7B"
)

# Convert table to a list of instruments
instruments <- tbl_redcap(conn) |>
  collect_list()

# Control labeling behavior
instruments_no_val_labels <- tbl_redcap(conn) |>
  collect_list(cols = FALSE)

# Convert coded values to text labels
instruments_with_codes <- tbl_redcap(conn) |>
  collect_list(convert = TRUE)

# Works with filtered data
filtered_instruments <- tbl_redcap(conn) |>
  filter(name_last == "Nutmouse") |>
  collect_list()

remove_duckdb(conn)

## End(Not run)
```

---

inspect *Inspect Project Data Table Structure*

Description

Inspects the structure of the project data table showing column information including name, type, and properties. This is a convenience wrapper around DBI::dbGetQuery() for examining table schema. Uses the data table name stored in the connection attributes if available.

Usage

```
inspect(conn, table_name = NULL)
```

Arguments

conn	A DuckDB connection object.
table_name	Character string specifying the table name. If NULL, uses the table name stored in connection attributes. Default is NULL.

Value

A data frame containing table information with columns for column ID, name, type, not null status, default value, and primary key.

Examples

```
## Not run:
table_info <- inspect(conn)

## End(Not run)
```

---

list_to_env *Assign List to Global Environment*

Description

Assign a list of instruments (data frames) to the global environment. Each element of the list becomes a separate object in the global environment.

Usage

```
list_to_env(instruments)
```

Arguments

instruments	A named list of data frames, typically output from tbl_to_list().

Value

Invisibly returns NULL. Side effect: assigns objects to the global environment.

Examples

```
## Not run:
tbl(conn, "data") |>
  tbl_to_list() |>
  list_to_env()

## End(Not run)
```

`metadata` *Get Metadata Table*

Description

Creates a tbl reference to the metadata table in the database and automatically collects it as a data frame. Uses the metadata table name stored in the connection attributes if available.

Usage

```
metadata(conn, metadata_table_name = NULL)
```

Arguments

`conn`	A DuckDB connection object.
`metadata_table_name`	Character string specifying the metadata table name. If NULL, uses the table name stored in connection attributes. Default is NULL.

Value

A data frame containing the metadata.

Examples

```
## Not run:
meta <- metadata(conn)

## End(Not run)
```

redcap_log *Get REDCap Logs Table*

Description

Creates a tbl reference to the REDCap logs table in the database and automatically collects it as a data frame. Uses the REDCap logs table name stored in the connection attributes if available.

Usage

```
redcap_log(conn, redcap_log_table_name = NULL)
```

Arguments

conn	A DuckDB connection object.
redcap_log_table_name	Character string specifying the REDCap logs table name. If NULL, uses the table name stored in connection attributes. Default is NULL.

Value

A data frame containing the REDCap audit log data.

Examples

```
## Not run:
redcap_log <- redcap_log(conn)

## End(Not run)
```

redcap_to_db *Transfer 'REDCap' Data to a Database*

Description

Transfer REDCap data to a database in chunks of record IDs to minimize memory usage.

Usage

```
redcap_to_db(
  conn,
  url,
  token,
  data_table_name = "data",
  metadata_table_name = "metadata",
```

```
  transfer_log_table_name = "transfer_log",
  redcap_log_table_name = "redcap_log",
  redcap_log_begin_date = Sys.Date() - 6,
  redcap_log_end_date = Sys.Date(),
  export_survey_fields = FALSE,
  export_data_access_groups = FALSE,
  blank_for_gray_form_status = FALSE,
  filter_logic = "",
  datetime_range_begin = as.POSIXct(NA),
  datetime_range_end = as.POSIXct(NA),
  fields = NULL,
  forms = NULL,
  events = NULL,
  record_id_name = "record_id",
  chunk_size = 1000,
  chunk_delay = 0.5,
  max_retries = 10,
  echo = "all",
  beep = TRUE,
  ...
)
```

Arguments

conn	A DBI connection object to a database.
url	Character string specifying the URI (uniform resource identifier) of the REDCap server's API.
token	Character string containing the REDCap API token specific to your project. This token is used for authentication and must have export permissions.
data_table_name	Character string specifying the name of the table to create or append data to. Default is "data". Can include schema name (e.g. "schema.table").
metadata_table_name	Character string specifying the name of the table to store REDCap metadata. Default is "metadata". Can include schema name (e.g. "schema.metadata").
transfer_log_table_name	Character string specifying the name of the table to store transfer logs. Default is "transfer_log". Can include schema name (e.g. "schema.transfer_log"). Set to NULL to disable logging.
redcap_log_table_name	Character string specifying the name of the table to store REDCap audit logs. Default is "redcap_log". Can include schema name (e.g. "schema.redcap_log"). Set to NULL to skip REDCap log retrieval.
redcap_log_begin_date	Date/POSIXct specifying the start date for REDCap log retrieval. Default is 6 days prior to today.

`redcap_log_end_date`
: Date/POSIXct specifying the end date for REDCap log retrieval. Default is today.

`export_survey_fields`
: Logical that specifies whether to export the survey identifier field (e.g., 'redcap_survey_identifier') or survey timestamp fields. Default is FALSE.

`export_data_access_groups`
: Logical that specifies whether or not to export the `redcap_data_access_group` field when data access groups are utilized in the project. Default is FALSE.

`blank_for_gray_form_status`
: Logical that specifies whether or not to export blank values for instrument complete status fields that have a gray status icon. Default is FALSE.

`filter_logic`
: String of logic text (e.g., `[gender] = 'male'`) for filtering the data to be returned, where the API will only return records where the logic evaluates as TRUE. Default is an empty string.

`datetime_range_begin`
: To return only records that have been created or modified *after* a given datetime, provide a POSIXct value. Default is NA (no begin time).

`datetime_range_end`
: To return only records that have been created or modified *before* a given datetime, provide a POSIXct value. Default is NA (no end time).

`fields`
: Character vector specifying which fields to export. Default is NULL (all fields).

`forms`
: Character vector specifying which forms to export. Default is NULL (all forms).

`events`
: Character vector specifying which events to export. Default is NULL (all events).

`record_id_name`
: Character string specifying the field name that contains record identifiers used for chunking requests. Default is "record_id".

`chunk_size`
: Integer specifying the number of record IDs per chunk. Default is 1000. Consider decreasing this for projects with many fields.

`chunk_delay`
: Numeric value specifying the delay in seconds between chunked requests. Default is 0.5 seconds. Adjust to respect REDCap server limits.

`max_retries`
: Integer specifying the maximum number of retry attempts for failed API connection or HTTP 504 error. Default is 10.

`echo`
: String to show a progress bar and all status messages ("all"), only a progress bar ("progress"), or nothing ("none"). Default is "all".

`beep`
: Logical indicating whether to play sound notifications when the process completes or encounters errors. Default is TRUE.

`...`
: Additional arguments passed to the REDCap API call.

Details

This function transfers data from REDCap to any database in chunks, which helps manage memory usage when dealing with large projects. It creates up to four tables in the database:

- `data_table_name`: Contains all transferred REDCap records
- `metadata_table_name`: Contains REDCap metadata for field definitions and labeling

- `transfer_log_table_name`: Contains timestamped logs of the transfer process
- `redcap_log_table_name`: Contains REDCap audit logs (optional)

The function automatically detects existing databases and handles them in three ways:

- If no table exists, starts a new transfer process
- If a table exists but is incomplete, resumes from the last processed record ID
- If a table exists and is complete, returns success without reprocessing

The function fetches record IDs, then processes records in chunks. If any error occurs during processing, the function will continue with remaining chunks but mark the transfer as incomplete.

If `redcap_log_table_name` is provided, the function will also retrieve REDCap audit logs and store them in a separate table. The date range for log retrieval can be controlled with `redcap_log_begin_date` and `redcap_log_end_date` parameters.

Data is first set to **VARCHAR/TEXT** type for consistent handling across chunks. For DuckDB, data types are automatically optimized after data is inserted:

- **INTEGER**: Columns with only whole numbers
- **DOUBLE**: Columns with decimal numbers
- **DATE**: Columns with valid dates
- **TIMESTAMP**: Columns with valid timestamps
- **VARCHAR/TEXT**: All other columns remain as strings

Value

Returns a list with the following components:

- `success`: Logical if the transfer was completed with no failed processing
- `error_chunks`: Vector of chunk numbers that failed processing
- `time_s`: Numeric value for total seconds to transfer and optimize data

See Also

`use_duckdb` for establishing a local duckdb connection `close_duckdb` for closing a local duckdb connection `remove_duckdb` for closing a local duckdb connection and removing the file `collect_labeled` for collecting a database table into a single data frame with column and value labels (converts coded values to their text labels by default) `collect_list` for collecting a database table into a list of instruments `collect_labeled_list` for collecting a database table into a list of instruments with column and value labels (converts coded values to their text labels by default) `list_to_env` for assigning a list of instruments to the global environment

Examples

```
## Not run:
library(redquack)
library(dplyr)

conn <- use_duckdb()
```

```
result <- redcap_to_db(
  conn,
  url = "https://bbmc.ouhsc.edu/redcap/api/",
  token = "9A81268476645C4E5F03428B8AC3AA7B"
)

data <- tbl_redcap(conn) |>
  collect()

remove_duckdb(conn)

## End(Not run)
```

`remove_duckdb` *Remove DuckDB Database*

Description

Closes a DuckDB connection and removes the database file.

Usage

```
remove_duckdb(conn, dbname = "redcap.duckdb")
```

Arguments

conn	A DuckDB connection object.
dbname	Character string specifying the database file name. Default is "redcap.duckdb".

Value

Invisible NULL.

Examples

```
## Not run:
conn <- use_duckdb()
# Use the connection...
remove_duckdb(conn)

## End(Not run)
```

save_parquet *Save Data to Parquet*

Description

Saves data directly from the database to a Parquet file using DuckDB's native COPY command. This is much faster than reading into R first and creates smaller files for easy sharing. Uses the data table name stored in the connection attributes if available.

Usage

```
save_parquet(
  conn,
  file_path = "redcap.parquet",
  table_name = NULL,
  query = NULL
)
```

Arguments

conn	A DuckDB connection object.
file_path	Character string specifying the output file path. Default is "redcap.parquet".
table_name	Character string specifying the source table name. If NULL, uses the table name stored in connection attributes. Default is NULL.
query	Character string with custom SQL query to export. If provided, table_name is ignored.

Value

Invisible NULL. Side effect: creates Parquet file at specified path.

Examples

```
## Not run:
# Save entire data table
save_parquet(conn, "redcap.parquet")

## End(Not run)
```

tbl_redcap *Create REDCap Data Table Reference*

Description

Creates a tbl reference to the main REDCap data table in the database. The original table name is stored as an attribute to preserve context through dplyr operations (limited to filter, select, arrange, and group_by). Uses the data table name stored in the connection attributes if available.

Usage

```
tbl_redcap(conn, table_name = NULL)
```

Arguments

conn	A DuckDB connection object.
table_name	Character string specifying the table name. If NULL, uses the table name stored in connection attributes. Default is NULL.

Value

A tbl_sql object referencing the data table with redcap_table attribute.

Examples

```
## Not run:
data <- tbl_redcap(conn)

## End(Not run)
```

transfer_log *Get Transfer Logs Table*

Description

Creates a tbl reference to the transfer logs table in the database and automatically collects it as a data frame. Uses the transfer logs table name stored in the connection attributes if available.

Usage

```
transfer_log(conn, transfer_log_table_name = NULL)
```

Arguments

conn	A DuckDB connection object.
transfer_log_table_name	Character string specifying the transfer logs table name. If NULL, uses the table name stored in connection attributes. Default is NULL.

Value

A data frame containing the transfer log data.

Examples

```
## Not run:
transfer_log <- transfer_log(conn)

## End(Not run)
```

---

use_duckdb *Create DuckDB Connection*

Description

Creates a DuckDB connection to the REDCap database file.

Usage

```
use_duckdb(dbname = "redcap.duckdb")
```

Arguments

dbname	Character string specifying the database file name. Default is "redcap.duckdb".

Value

A DuckDB connection object.

Examples

```
## Not run:
conn <- use_duckdb()
# Use the connection...
remove_duckdb(conn)

## End(Not run)
```

Index