Package ‘ctrdata’

October 12, 2022

Type Package

Title Retrieve and Analyze Clinical Trials in Public Registers

Version 1.10.2

Imports jsonlite, httr, curl, clipr, xml2, rvest, nodbi (>= 0.7.0), stringi, dplyr, lubridate

SystemRequirements sed, php, cat, perl

URL https://cran.r-project.org/package=ctrdata

BugReports https://github.com/rfhb/ctrdata/issues


Trial information is downloaded, converted and stored in a database ('PostgreSQL', 'SQLite' or 'MongoDB'; via package 'nodbi'). Functions are included to identify deduplicated records, to easily find and extract variables (fields) of interest even from complex nesting as used by the registers, and to update previous queries.

The package can be used for meta-analysis and trend-analysis of the design and conduct as well as for results of clinical trials.

License MIT + file LICENSE

RoxygenNote 7.2.1

Suggests devtools, knitr, rmarkdown, RSQLite (>= 2.2.4), mongolite, tinytest (>= 1.2.1), R.rsp

VignetteBuilder R.rsp

NeedsCompilation no

Encoding UTF-8

Author Ralf Herold [aut, cre] (<https://orcid.org/0000-0002-8148-6748>)

Maintainer Ralf Herold <ralf.herold@mailbox.org>
A package for aggregating and analysing information on and results from clinical trials, retrieved from public study registers

Database connection
Package ctrdata retrieves trial information and stores it in a database collection, which has to be given as a connection object to parameter \texttt{con} for several \texttt{ctrdata} functions; this connection object is created in slightly different ways for the three supported database backends:

<table>
<thead>
<tr>
<th>Database</th>
<th>Connection object</th>
</tr>
</thead>
<tbody>
<tr>
<td>MongoDB</td>
<td>\texttt{dbc &lt;- nodbi::src_mongo(db = &quot;my_db&quot;, collection = &quot;my_coll&quot;)}</td>
</tr>
<tr>
<td>SQLite</td>
<td>\texttt{dbc &lt;- nodbi::src_sqlite(dbname = &quot;my_db&quot;, collection = &quot;my_coll&quot;)}</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>\texttt{dbc &lt;- nodbi::src_postgres(dbname = &quot;my_db&quot;); dbc[&quot;collection&quot;] &lt;- &quot;my_coll&quot;}</td>
</tr>
</tbody>
</table>

Example of using a \texttt{ctrdata} function with any such connection object: \texttt{ctrdata::dbQueryHistory(con = \texttt{dbc})}. Besides \texttt{ctrdata} functions below, any such a connection object can equally be used with functions of package nodbi, for example \texttt{nodbi::docdb_query(src = \texttt{dbc}, key = \texttt{dbc$collection}, fields = '{"_id": 1}', query = '{"sponsors.lead_sponsor.agency_class": "Industry"}')
Operations on a clinical trial register

ctrOpenSearchPagesInBrowser, ctrLoadQueryIntoDb (load trial records into database collection), ctrFindActiveSubstanceSynonyms; see ctrdata-registers for details on registers and how to search

Get a data frame from the database collection

dbFindFields (find names of fields of interest in trial records in a collection), dbGetFieldsIntoDf (create a data frame for fields of interest from collection), dbFindIdsUniqueTrials (de-duplicated identifiers of clinical trial records that can be used to subset a data frame)

Operate on a data frame with trial information

dfTrials2Long (convert fields with nested elements into long format), dfName2Value (get values for variable(s) of interest), dfMergeTwoVariablesRelevel

Author(s)

Ralf Herold <ralf.herold@mailbox.org>

ctrdata-registers        ctrdata: detailed information on clinical trial registers

Description

Registers of clinical trials that can be accessed with package ctrdata-package as of end 2021

Details

- **EUCTR**: The European Union Clinical Trials Register contains more than 40,000 clinical trials (using one or more medicines as investigational medicinal product, IMP; in Europe and beyond)
- **CTGOV**: ClinicalTrials.gov includes almost 400,000 interventional and observational studies
- **ISRCTN**: The ISRCTN Registry includes more than 21,000 interventional or observational health studies

<table>
<thead>
<tr>
<th>Material</th>
<th>EUCTR</th>
<th>CTGOV</th>
<th>ISRCTN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home page</td>
<td>link</td>
<td>link</td>
<td>link</td>
</tr>
<tr>
<td>About</td>
<td>link</td>
<td>link</td>
<td>link</td>
</tr>
<tr>
<td>Terms and conditions, disclaimer</td>
<td>link</td>
<td>link</td>
<td>link</td>
</tr>
<tr>
<td>How to search</td>
<td>link</td>
<td>link</td>
<td>link</td>
</tr>
<tr>
<td>Search interface</td>
<td>link</td>
<td>link</td>
<td>link</td>
</tr>
<tr>
<td>Glossary</td>
<td>link</td>
<td>link</td>
<td>link</td>
</tr>
<tr>
<td>FAQ</td>
<td>link</td>
<td>link</td>
<td>link</td>
</tr>
<tr>
<td>Expert / advanced search</td>
<td>link</td>
<td>link</td>
<td>link</td>
</tr>
<tr>
<td>Example*</td>
<td>link</td>
<td>link</td>
<td>link</td>
</tr>
<tr>
<td>Definitions</td>
<td>link</td>
<td>Protocol, results, names, syntax</td>
<td>link</td>
</tr>
</tbody>
</table>
*The example expert search retrieves interventional trials with neonates, investigating infectious conditions: EUCTR retrieves trials with neonates, but not exclusively. The CTGOV expert search retrieves trials exclusively in neonates. ISRCTN retrieves a small number of studies. Thus, after loading trials with `ctrLoadQueryIntoDb` into a database collection, corresponding sets of trials need to be defined, based on values of fields of interest (e.g., `eligibility.maximun_age` from CTGOV and `f115_childen_211years` from EUCTR), which can be obtained from the collection using `dbGetFieldsIntoDf`.

Author(s)

Ralf Herold <ralf.herold@mailbox.org>

---

### ctrFindActiveSubstanceSynonyms

**Find synonyms of an active substance**

### Description

An active substance can be identified by a recommended international nonproprietary name (INN), a trade or product name, or a company code(s). Retrieves the substances which are searched for by the register 'ClinicalTrials.Gov' for a given active substance.

### Usage

```r
ctrFindActiveSubstanceSynonyms(activesubstance = "")
```

### Arguments

- **activesubstance**
  An active substance, in an atomic character vector

### Value

A character vector of the active substance (input parameter) and synonyms, or NULL if active substance was not found and may be invalid

### Examples

```r
## Not run:
ctrFindActiveSubstanceSynonyms(activesubstance = "imatinib")
# [1] "imatinib" "gleevec" "sti 571" "glivec" "CGP 57148" "st1571"

## End(Not run)
```
ctrGetQueryUrl

Get query details

Description
Extracts query parameters and register name from parameter `url` or from the clipboard, into which the URL of a register search was copied.

Usage

```
ctrGetQueryUrl(url = '"", register = '"
```

Arguments

- **url**: URL such as from the browser address bar. If not specified, clipboard contents will be checked for a suitable URL. Can also contain a query term such as from `dbQueryHistory()`["query-term"]
- **register**: Optional name of register (i.e., "EUCTR" or "CTGOV") in case url is a query term

Value
A data frame (or tibble, if `dplyr` is loaded) with column names `query-term` and `query-register`. The data frame (or tibble) can be passed as such as parameter `query-term` to `ctrLoadQueryIntoDb` and as parameter `url` to `ctrOpenSearchPagesInBrowser`.

Examples

```
# user copied into the clipboard the URL from
# the address bar of the browser that shows results
# from a query in one of the trial registers
try(ctrGetQueryUrl(), silent = TRUE)

# extract query parameters from search result URL
# (URL was cut for the purpose of formatting only)
ctrGetQueryUrl(
  url = paste0("https://clinicaltrials.gov/ct2/results?",
  "cond=&term=AREA%5BMaximumAge%5D+RANGE%5B0+days%2C+28+days%5D",
  "&type=Intr&rslt=&age_v=&gnrd=&intr=Drugs%2C+Investigational",
  "&titles=&outc=&spons=&lead=&id=&cntry=&state=&city=&dist=",
  "&locn=&phase=2&rsub=&strd_s=01%2F01%2F2015&strd_e=01%2F01%2F2016",
  "&prcd_s=&prcd_e=&sfpd_s=&sfpd_e=&rfpd_s=&rfpd_e=&lupd_s=&lupd_e=&sort="
))
```
ctrLoadQueryIntoDb  
*Load and store register trial information*

**Description**

Retrieves or updates information on clinical trials from registers and stores it in a collection in a database. This is the main function of ctrdata-package for accessing registers and loading trial information into a database collection, even if from different queries or different registers. The query details are stored in the database collection and can be accessed using dbQueryHistory. A previous query can be re-run, which replaces or adds trial records. However, user annotations of trial records are kept.

**Usage**

```r
ctrLoadQueryIntoDb(
  queryterm = NULL,
  register = "",
  querytoupdate = NULL,
  forctoupdate = FALSE,
  euctrresults = FALSE,
  euctrresultshistory = FALSE,
  euctrresultsfilepath = euctrresultspdfpath,
  euctrresultspdfpath = NULL,
  annotation.text = "",
  annotation.mode = "append",
  parallelretrievals = 10L,
  only.count = FALSE,
  con = NULL,
  verbose = FALSE
)
```

**Arguments**

- **queryterm**: Either a string with the full URL of a search in a register, or the data frame returned by the ctrGetQueryUrl or the dbQueryHistory functions, or, together with parameter register, a string with query elements of a search URL. The queryterm is recorded in the collection for later use to update records.
- **register**: String with abbreviation of register to query, either "EUCTR", "CTGOV" or "ISRCTN". Not needed if queryterm provide the information which register to query (see queryterm).
- **querytoupdate**: Either the word "last" or the number of the query (based on dbQueryHistory) that should be run to retrieve any trial records that are new or have been updated since this query was run the last time. This parameter takes precedence over queryterm. For EUCTR, updates are available only for the last seven days; the query is run again if more time has passed since it was run last.
forcetoupdate If TRUE, run again the query given in querytoupdate, irrespective of when it
was run last (default is FALSE).
euctrresults If TRUE, also download available results when retrieving and loading trials from
EUCTR. This slows down this function. (For CTGOV, all available results are
always retrieved and loaded.)
euctrrresultshistory If TRUE, also download available history of results publication in EUCTR. This
is quite time-consuming (default is FALSE).
euctrresultsfiles诸多 If this is a relative or absolute path to a directory that exists or can be created,
save results files into it, e.g., PDF files of result publications that had been sub-
mited to EUCTR (default is NULL).
euctrresultspdfpath Deprecated, use euctrresultsfilespath
annotation.text Text to be including in the records retrieved with the current query, in the field
"annotation".
annotation.mode One of "append" (default), "prepend" or "replace" for new annotation.text with
respect to any existing annotation for the records retrieved with the current
query.
parallelretrievals Number of parallel downloads of information from the register, defaults to 10.
only.count Set to TRUE to return only the number of trial records found in the register for
the query. Does not load trial information into the database. Default is FALSE.
con A connection object, see section 'Databases' in ctrdata-package
verbose Printing additional information if set to TRUE; default is FALSE.

Value
A list with elements 'n' (number of trial records newly imported or updated), 'success' (a vector
of _id's of successfully loaded records), 'failed' (a vector of identifiers of records that failed to
load) and 'queryterm' (the query term used). The returned list has several attributes (including
database and collection name, as well as the query history of this database collection) to facilitate
documentation.

Examples
## Not run:

dbc <- nodbi::src_sqlite(collection = "my_collection")

# Retrieve protocol- and results-related information
# on a single trial identified by its EU number
ctrLoadQueryIntoDb(
  queryterm = "2013-001291-38",
  register = "EUCTR",

euctrresults = TRUE,
con = dbc
)

# Retrieve all information on about 2,000 ongoing
# interventional cancer trials involving children
# into the same collection as used before
ctrLoadQueryIntoDb(
    queryterm = "cancer&recr=Open&type=Intr&age=0",
    register = "CTGOV",
    con = dbc
)

# Retrieve all information on more than 40 trials
# that are labelled as phase 3 and that mention
# either neuroblastoma or lymphoma from ISRCTN,
# into the same collection as used before
ctrLoadQueryIntoDb(
    queryterm = "https://www.isrctn.com/search?q=neuroblastoma+OR+lymphoma&filters=phase%3APhase+III",
    con = dbc
)

## End(Not run)

ctrOpenSearchPagesInBrowser

_Open stored query or register search page_

**Description**

Open advanced search pages of register(s), or execute search in browser

**Usage**

ctrOpenSearchPagesInBrowser(url = "", register = "", copyright = FALSE, ...)

**Arguments**

- **url**
  - of search results page to show in the browser. Can be the return value of `ctrGetQueryUrl` or `dbQueryHistory`.

- **register**
  - Register(s) to open. Either "EUCTR" or "CTGOV" or a vector of both. Default is to open both registers’ advanced search pages. To open the browser with a previous search, the output of `ctrGetQueryUrl` or one row from `dbQueryHistory` can be used.

- **copyright**
  - (Optional) If set to TRUE, opens copyright pages of register(s).

- **...**
  - May include the deprecated input parameter.
**dbFindFields**

**Value**

Always TRUE, invisibly.

**Examples**

```r
# Check copyrights before using registers
crOpenSearchPagesInBrowser(copyright = TRUE)

# open all queries loaded into demo collection
dbc <- nodbi::src_sqlite(
  dbname = system.file("extdata", "demo.sqlite", package = "ctrdata"),
  collection = "my_trials")

dbh <- dbQueryHistory(con = dbc)

for (r in seq_len(nrow(dbh))) {
  ctrOpenSearchPagesInBrowser(dbh[r, ])
}
```

**dbFindFields**

Find names of fields in the database collection

**Description**

Given part of the name of a field of interest to the user, this function returns the full field names used in records that were previously loaded into a collection (using `ctrLoadQueryIntoDb`). The field names can be fed into function `dbGetFieldsIntoDf` to extract the data from the collection into a data frame. In addition to the full names of leaf fields (e.g., `clinical_results.outcome_list.outcome.measure.class_list.class.title`) this function also returns names of node fields (e.g., `clinical_results`). Data in node fields is typically complex (multiply nested) and can be converted into individual data elements by function `dfTrials2Long`, possibly followed by function `dfName2Value`.

**Usage**

```r
dbFindFields(namepart = "", con, verbose = FALSE)
```

**Arguments**

- `namepart` A plain string (can include a regular expression, including Perl-style) to be searched for among all field names (keys) in the collection. Use `.\.*` to find all fields.
- `con` A connection object, see section ‘Databases’ in ctrdata-package
- `verbose` If TRUE, prints additional information (default FALSE).
dbFindIdsUniqueTrials

Details


For fields in CTGOV (protocol-related information), see https://prsinfo.clinicaltrials.gov/definitions.html.

For fields in ISRCTN (protocol-related information), see https://www.isrctn.com/page/definitions.

Note: Only when ‘dbFindFields’ is first called after ctrLoadQueryIntoDb, it will take a moment.

Value

Vector of strings with full names of field(s) found, in alphabetical order by register. This is a named vector where the names of the vector are the register names for the respective fields.

Examples

```
dbc <- nodbi::src_sqlite(
  dbname = system.file("extdata", "demo.sqlite", package = "ctrdata"),
  collection = "my_trials")

dbFindFields(namepart = "date", con = dbc)
```

dbFindIdsUniqueTrials Get identifiers of deduplicated trial records

Description

Records for a clinical trial can be loaded from more than one register into a collection. The function returns identifiers of records in the collection that were loaded from the register(s) preferred by the user. All registers are recording identifiers also from other registers, which are used by this function to provide a vector of identifiers of deduplicated trials.

Usage

```
dbFindIdsUniqueTrials(
  preferregister = c("EUCTR", "CTGOV", "ISRCTN"),
  prefermemberstate = "DE",
  include3rdcountrytrials = TRUE,
  con,
  verbose = FALSE
)
```
**Arguments**

- `preferregister` A vector of the order of preference for registers from which to generate unique _id’s, default `c("EUCTR", "CTGOV", "ISRCTN")`
- `prefermemberstate` Code of single EU Member State for which records should returned. If not available, a record for DE or lacking this, any random Member State’s record for the trial will be returned. For a list of codes of EU Member States, please see vector `countriesEUCTR`. Specifying "3RD" will return the Third Country record of trials, where available.
- `include3rdcountrytrials` A logical value if trials should be retained that are conducted exclusively in third countries, that is, outside the European Union. Ignored if `prefermemberstate` is set to "3RD".
- `con` A connection object, see section ‘Databases’ in ctrdata-package
- `verbose` If set to TRUE, prints out which fields of the registers are used to find corresponding trial records

**Details**

Note that the content of records may differ between registers (and, for EUCTR, between records for different Member States). Such differences are not considered by this function.

**Value**

A vector with strings of keys ("_id") of records in the collection that represent unique trials.

**Examples**

```r
dbc <- nodbi::src_sqlite(
  dbname = system.file("extdata", "demo.sqlite", package = "ctrdata"),
  collection = "my_trials")

dbFindIdsUniqueTrials(con = dbc)
```

---

**Description**

Fields in the collection are retrieved into a data frame (or tibble). Note that fields within the record of a trial can be hierarchical and structured, that is, nested. Names of fields can be found with `dbFindFields`. The function uses the field names to appropriately type the values that it returns, harmonising original values (e.g. "Information not present in EudraCT" becomes ‘NA’, "Yes" becomes ‘TRUE’, "false" becomes ‘FALSE’, date strings become class Date, number strings become
numbers). The function attempts so simplify the structure of some nested data and may concatenate multiple strings in a field using " / " (see below); for complex nested data, use function dfTrials2Long followed by dfName2Value to extract the desired nested variable(s).

Usage

\[
\text{dbGetFieldsIntoDf}(\text{fields} = "", \text{con}, \text{verbose} = \text{FALSE}, \text{stopifnodata} = \text{TRUE})
\]

Arguments

- **fields**: Vector of one or more strings, with names of sought fields. See function dbFindFields for how to find names of fields. “item.subitem” notation is supported.
- **con**: A connection object, see section ‘Databases’ in ctrdata-package
- **verbose**: Printing additional information if set to TRUE; (default FALSE).
- **stopifnodata**: Stops with an error (default TRUE) or with a warning (FALSE) if the sought field is empty in all, or not available in any of the records in the database collection.

Value

A data frame (or tibble, if dplyr is loaded) with columns corresponding to the sought fields. A column for the record ‘_id’ will always be included. Each column can be either a simple data type (numeric, character, date) or a list. For complicated lists, use function dfTrials2Long followed by function dfName2Value to extract values for nested variables. The maximum number of rows of the returned data frame is equal to, or less than the number of records of trials in the database collection.

Examples

\[
\text{dbc <- nodbi::src_sqlite(}
\quad \text{dbname = system.file("extdata", "demo.sqlite", package = "ctrdata"),}
\quad \text{collection = "my_trials"})
\]

# get fields that are nested within another field
# and can have multiple values with the nested field
\text{dbGetFieldsIntoDf(}
\quad \text{fields = "b1_sponsor.b31_and_b32_status_of_the_sponsor",}
\quad \text{con = dbc})

# fields that are lists of string values are
# returned by concatenating values with a slash
\text{dbGetFieldsIntoDf(}
\quad \text{fields = "keyword",}
\quad \text{con = dbc})
dbQueryHistory

Show history of queries loaded into a database collection

Description

Show history of queries loaded into a database collection

Usage

dbQueryHistory(con, verbose = FALSE)

Arguments

con               A connection object, see section ‘Databases’ in ctrdata-package
verbose           If TRUE, prints additional information (default FALSE).

Value

A data frame (or tibble, if dplyr is loaded) with columns: ‘query-timestamp’, ‘query-register’,
‘query-records’ (note: this is the number of records loaded when last executing ctrLoadQueryIntoDb,
not the total record number) and ‘query-term’, with one row for each time ctrLoadQueryIntoDb
loaded trial records into this collection.

Examples

```r
dbc <- nodbi::src_sqlite(
  dbname = system.file("extdata", "demo.sqlite", package = "ctrdata"),
  collection = "my_trials")

dbQueryHistory(con = dbc)
```

dfListExtractKey

Extract named element(s) from list(s) into long-format data frame

Description

(Deprecated, use dfTrials2Long and dfName2Value!) The function uses a name (key) to extract an
element from a list in a data.frame such as obtained with dbGetFieldsIntoDf. This helps to simplify
working with nested lists and with complex structures.

Usage

```
dfListExtractKey(df, list.key = list(c("endPoints.endPoint", "^title")))
```
dfMergeTwoVariablesRelevel

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>df</td>
<td>A data frame (or tibble)</td>
</tr>
<tr>
<td>list.key</td>
<td>A list of pairs of list names and key names, where the list name corresponds to the name of a column in df that holds a list and the name of the key identifies the element to be extracted. See example.</td>
</tr>
</tbody>
</table>

Value

A data frame (or tibble, if dplyr is loaded) in long format with columns name (identifying the full path in the data frame, "<list>.<key>"), _id (of the trial record), value (of name per _id), item (number of value of name per _id).

Examples

```r
dbc <- nodbi::src_sqlite(
  dbname = system.file("extdata", "demo.sqlite", package = "ctrdata"),
  collection = "my_trials")

df <- dbGetFieldsIntoDf(
  fields = c("endPoints.endPoint",
              "subjectDisposition.postAssignmentPeriods"),
  con = dbc)

suppressWarnings(
  dfListExtractKey(
    df = df,
    list.key = list(
      c("endPoints.endPoint", "^title"),
      c("subjectDisposition.postAssignmentPeriods", "arms.arm.type.value")))
)
```

---

**dfMergeTwoVariablesRelevel**

*Merge two variables*

Description

Merge two variables in a data frame such as returned by dbGetFieldsIntoDf into a new variable, and optionally also map its values to new levels.

Usage

```
dfMergeTwoVariablesRelevel(df = NULL, colnames = "", levelslist = NULL, ...)
```
Arguments

df \hspace{1cm} \text{A data frame in which there are two variables (columns) to be merged into one.}

colnames \hspace{1cm} \text{A vector of length two with names of the two columns that hold the variables to be merged. See \texttt{colnames} for how to obtain the names of columns of a data frame.}

levelslist \hspace{1cm} \text{A list with one slice each for a new value to be used for a vector of old values (optional).}

... \hspace{1cm} \text{for deprecated varnames parameter (will be removed)}

Value

A vector of strings

Examples

defvars2merge <- c("overall_status", "x5_trial_status")

dbc <- nodbi::src_sqlite(
  dbname = system.file("extdata", "demo.sqlite", package = "ctrdata"),
  collection = "my_trials")

df <- dbGetFieldsIntoDf(
  fields = vars2merge,
  con = dbc)

statusvalues <- list(
  "ongoing" = c("Recruiting", "Active", "Ongoing",
                "Active, not recruiting", "Enrolling by invitation"),
  "completed" = c("Completed", "Prematurely Ended", "Terminated"),
  "other" = c("Withdrawn", "Suspended",
               "No longer available", "Not yet recruiting"))

dfMergeTwoVariablesRelevel(
  df = df,
  colnames = vars2merge,
  levelslist = statusvalues)

\begin{center}
\begin{tabular}{ll}
\textbf{dfName2Value} & \textit{Get value for variable of interest} \\
\end{tabular}
\end{center}

Description

Get information of interest (e.g., endpoint) from long data frame of protocol- or result-related trial information as returned by \texttt{dfTrials2Long}. Parameters ‘valuename’, ‘wherename’ and ‘wherevalue’ are matched using Perl regular expressions and ignoring case.
Usage

```r
dfName2Value(df, valuename = "", wherename = "", wherevalue = "")
```

Arguments

- **df**: A data frame (or tibble) with four columns ('_id', 'identifier', 'name', 'value') as returned by `dfTrials2Long`
- **valuename**: A character string for the name of the field that holds the value of the variable of interest (e.g., a summary measure such as "endPoints.*tendencyValue.value")
- **wherename**: (optional) A character string to identify the variable of interest among those that repeatedly occur in a trial record (e.g., "endPoints.endPoint.title")
- **wherevalue**: (optional) A character string with the value of the variable identified by 'wherename' (e.g., "response")

Value

A data frame (or tibble, if `dplyr` is loaded) that includes the values of interest, with columns '_id', 'identifier', 'name', 'value' (and 'where', with the contents of 'wherevalue' found at 'wherename'). Contents of 'value' are strings unless all its elements are numbers. The 'identifier' is generated by function `dfTrials2Long` to identify matching elements, e.g. endpoint descriptions and measurements.

Examples

```r
dbc <- nodbi::src_sqlite(
  dbname = system.file("extdata", "demo.sqlite", package = "ctrdata"),
  collection = "my_trials")

dfwide <- dbGetFieldsIntoDf(
  fields = c(
    ## ctgov - typical results fields
    # "clinical_results.baseline.analyzed_list.analyzed.count_list.count",
    # "clinical_results.baseline.group_list.group",
    # "clinical_results.outcome_list.outcome",
    # "study_design_info.allocation",
    ## euctr - typical results fields
    # "trialInformation.fullTitle",
    # "baselineCharacteristics.baselineReportingGroups.baselineReportingGroup",
    # "trialChanges.hasGlobalInterruptions",
    # "subjectAnalysisSets",
    # "adverseEvents.seriousAdverseEvents.seriousAdverseEvent",
    "endPoints.endPoint",
    "subjectDisposition.recruitmentDetails" ), con = dbc)

dfLong <- dfTrials2Long(df = dfwide)

## get values for the endpoint 'response'
```
dfTrials2Long

Convert data frame with trial records into long format

Description

The function works with protocol- and results-related information. It converts lists and other values that are in a data frame returned by dbGetFieldsIntoDf into individual rows of a long data frame. From the resulting data frame, values of interest can be selected using dfName2Value. The function is intended for fields with complex content, such as node field "clinical_results" from EUCTR, which dbGetFieldsIntoDf returns as a multiply nested list and for which this function then converts every observation of every (leaf) field into a row of its own.

Usage

dfTrials2Long(df)

Arguments

df

Data frame (or tibble) with columns including the trial identifier (_id) and one or more variables as obtained from dbGetFieldsIntoDf

Value

A data frame (or tibble, if dplyr is loaded) with the four columns: '_id', 'identifier', 'name', 'value'

Examples

dbc <- nodbi::src_sqlite(
  dbname = system.file("extdata", "demo.sqlite", package = "ctrdata"),
  collection = "my_trials")

dfwide <- dbGetFieldsIntoDf(
  fields = "clinical_results.participant_flow",
  con = dbc)
dfTrials2Long(df = dfwide)

installCygwinWindowsDoInstall

Install necessary helper apps (Windows only)

Description

Convenience function to install a minimal Cygwin environment under MS Windows, including perl, sed and php. Alternatively and in case of difficulties, download and run the cygwin setup yourself as follows:


These binaries are required only for function **ctr-LoadQueryIntoDb** but not for any other function in this package.

Usage

installCygwinWindowsDoInstall(force = FALSE, proxy = Sys.getenv("https_proxy"))

Arguments

- **force**
  
  Set to **TRUE** to update a Cygwin environment that was previously installed with the function, or to overwrite any existing installation in **c:\cygwin**

- **proxy**
  
  Specify any proxy to be used for downloading via http, e.g. `host_or_ip:port`; defaults to the environment variable `https_proxy`. Set to `""` to not specify or to unset a proxy.

Examples

```r
## Not run:

try(installCygwinWindowsDoInstall(), silent = TRUE)
```

## End(Not run)
Index

* data
  * ctrdata-registers, 3
* package
  * ctrdata-package, 2

colnames, 15
ctrdata-package, 2, 3, 6, 7, 9, 11–13
ctrdata-registers, 3, 3
ctrdata:::dbQueryHistory, 2
ctrFindActiveSubstanceSynonyms, 3, 4
ctrGetQueryUrl, 5, 6, 8
ctrLoadQueryIntoDb, 3–5, 6, 9, 10, 13, 18
ctrOpenSearchPagesInBrowser, 3, 5, 8
data.frame, 15
dbFindFields, 3, 9, 11, 12
dbFindIdsUniqueTrials, 3, 10
dbGetFieldsIntoDf, 3, 4, 9, 11, 13, 14, 17
dbQueryHistory, 5, 6, 8, 13
dfListExtractKey, 13
dfMergeTwoVariablesRelevel, 3, 14
dfName2Value, 3, 9, 12, 13, 15, 17
dfTrials2Long, 3, 9, 12, 13, 15, 16, 17
installCygwinWindowsDoInstall, 18

nodbi:::docdb_query, 2
nodbi:::src_mongo, 2
nodbi:::src_postgres, 2
nodbi:::src_sqlite, 2