

Package ‘ctrdata’

November 3, 2020

Type Package

Title Retrieve and Analyze Clinical Trials in Public Registers

Version 1.4.1

Date 2020-11-03

Imports jsonlite, httr, curl, clipr, xml2, rvest, nodbi (>= 0.4)

SystemRequirements sed, php, cat, perl

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

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

Description Provides functions for querying, retrieving and analyzing protocol- and results-related information on clinical trials from two public registers, the 'European Union Clinical Trials Register' ('EUCTR', <<https://www.clinicaltrialsregister.eu/>>) and 'ClinicalTrials.gov' ('CTGOV', <<https://clinicaltrials.gov/>>). The trial information is transformed and stored in a database ('SQLite' or 'MongoDB', via 'nodbi'). Functions are provided to identify de-duplicated records, to easily find and extract variables (fields) of interest even from complex nesting as used by the registers, and to update previous queries that users retrieved in a database. The package can be used for meta analysis and trend-analysis of the design and conduct as well as results of clinical trials.

License MIT + file LICENSE

RoxygenNote 7.1.1

LazyData true

Suggests devtools, knitr, rmarkdown, RSQLite (>= 2.1.2), 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>

Repository CRAN

Date/Publication 2020-11-03 21:30:03 UTC

R topics documented:

ctrFindActiveSubstanceSynonyms	2
ctrGetQueryUrlFromBrowser	3
ctrLoadQueryIntoDb	4
ctrOpenSearchPagesInBrowser	6
dbFindFields	7
dbFindIdsUniqueTrials	8
dbGetFieldsIntoDf	9
dbQueryHistory	10
dfListExtractKey	11
dfMergeTwoVariablesRelevel	12
dfName2Value	13
dfTrials2Long	14
installCygwinWindowsDoInstall	15
Index	17

ctrFindActiveSubstanceSynonyms

Find synonyms of an active substance

Description

An active substance can be identified by a recommended international nonproprietary name, a trade or product name, or a company code(s).

Usage

```
ctrFindActiveSubstanceSynonyms(activessubstance = "")
```

Arguments

activessubstance

An active substance, in an atomic character vector

Details

At this time, this function uses the register ClinicalTrials.Gov to detect which substances were also searched for.

Value

A character vector of the active substance (input parameter) and synonyms, if any were found

Examples

```
## Not run:
ctrFindActiveSubstanceSynonyms(
  activesubstance = "imatinib"
)

## End(Not run)
```

```
ctrGetQueryUrlFromBrowser
```

Import from clipboard the URL of a search in one of the registers

Description

Import from clipboard the URL of a search in one of the registers

Usage

```
ctrGetQueryUrlFromBrowser(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 string of query parameters that can be used to retrieve data from the register.

A data frame with a query term and the register name that can directly be used in [ctrLoadQueryIntoDb](#) and in [ctrOpenSearchPagesInBrowser](#)

Examples

```
## Not run:
db <- nodbi::src_sqlite(
  collection = "my_collection"
)
ctrLoadQueryIntoDb(
  ctrGetQueryUrlFromBrowser(),
  con = db
)
```

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

ctrLoadQueryIntoDb	<i>Retrieve or update information on clinical trials from register and store in database</i>
--------------------	--

Description

This is the main function of package `ctrdata` for accessing registers. Note that re-running this function adds or updates trial records in a database, even if from different queries or different registers.

Usage

```
ctrLoadQueryIntoDb(
  queryterm = "",
  register = "EUCTR",
  querytoupdate = 0L,
  forcetoupdate = FALSE,
  ectrresults = FALSE,
  ectrresultshistory = FALSE,
  annotation.text = "",
  annotation.mode = "append",
  parallelretrievals = 10L,
  only.count = FALSE,
  con,
  verbose = FALSE
)
```

Arguments

queryterm	Either a string with the full URL of a search in a register or the data frame returned by the <code>ctrGetQueryUrlFromBrowser</code> or the <code>dbQueryHistory</code> functions. The queryterm is recorded in the collection for later use to update records.
register	Vector of abbreviations of the register to query, defaults to "EUCTR".
querytoupdate	Either the word "last" or the number of the query (based on <code>dbQueryHistory</code>) 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).
ectrresults	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.)

euctrresultshistory	If TRUE, also download available history of results publication in EUCTR. This is quite time-consuming (default is FALSE).
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 src connection object, as obtained with <code>nodbi::src_mongo()</code> or <code>nodbi::src_sqlite()</code>
verbose	Printing additional information if set to TRUE; default is FALSE.

Value

A list with elements "n" (the number of trials that were newly imported or updated with this function call), "ids" (a vector of the `_id[s]` of these trials) and the "queryterm" used, with several attributes set (database connection details and a data frame of the query history in this database).

Examples

```
## Not run:
db <- nodbi::src_sqlite(
  collection = "test"
)
# Retrieve protocol-related information on a
# single trial identified by EudraCT number
ctrLoadQueryIntoDb(
  queryterm = "2013-001291-38", con = db
)
# Retrieve protocol-related information on
# ongoing interventional cancer trials in children
ctrLoadQueryIntoDb(
  queryterm = "cancer&recr=Open&type=Intr&age=0",
  register = "CTGOV",
  con = db
)

## End(Not run)
```

ctrOpenSearchPagesInBrowser

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

Description

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

Usage

```
ctrOpenSearchPagesInBrowser(input = "", register = "", copyright = FALSE, ...)
```

Arguments

input	Show results of search for queryterm in browser. To open the browser with a previous search, (register or) queryterm can be the output of ctrGetQueryUrlFromBrowser or can be one row from 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 ctrGetQueryUrlFromBrowser() or one row from dbQueryHistory() can be used.
copyright	(Optional) If set to TRUE, opens copyright pages of register(s).
...	Any additional parameter to use with browseURL, which is called by this function.

Value

Is always true, invisibly.

Examples

```
## Not run:
ctrOpenSearchPagesInBrowser(
  "https://www.clinicaltrialsregister.eu/ctr-search/search?query=cancer")

ctrOpenSearchPagesInBrowser(
  ctrGetQueryUrlFromBrowser())

# open the last query that was
# loaded into the database
db <- nodbi::src_sqlite(
  collection = "previously_created"
)
ctrOpenSearchPagesInBrowser(
  dbQueryHistory(con = db))

## End(Not run)
```

dbFindFields	<i>Find names of fields in the database collection</i>
--------------	--

Description

Given part of the name of a field of interest to the user, this function returns the full field names as found in the database.

Usage

```
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 database.
con	A <code>src</code> connection object, as obtained with <code>nodbi::src_mongo()</code> or <code>nodbi::src_sqlite()</code>
verbose	If TRUE, prints additional information (default FALSE).

Details

For fields in EUCTR (protocol- and results-related information), see also the register's documentation at <https://eudract.ema.europa.eu/>.

For fields in CTGOV (protocol-related information), see also the register's definitions at <https://prsinfo.clinicaltrials.gov/definitions.html>.

Note: Generating a list of fields with this function may take some time, and may involve running a mapreduce function is run on the server. If the user is not not authorized to run such a function on the (local or remote) server, random documents are sampled to generate a list of fields.

Value

Vector of names of found field(s)

Examples

```
## Not run:
db <- nodbi::src_sqlite(
  collection = "my_collection"
)
dbFindFields(
  namepart = "date",
  con = db
)

## End(Not run)
```

dbFindIdsUniqueTrials *De-duplicate records to provide unique clinical trial identifiers*

Description

If records for a clinical trial are found from more than one register, the record from EUCTR is returned. The function currently relies on CTGOV recording other identifiers such as the EudraCT number in the field "Other IDs".

Usage

```
dbFindIdsUniqueTrials(
  preferregister = "EUCTR",
  prefermemberstate = "GB",
  include3rdcountrytrials = TRUE,
  con,
  verbose = TRUE
)
```

Arguments

preferregister The abbreviation of the preferred register, in case a trial is in more than one register (string, either "EUCTR" or "CTGOV"). If set to an empty string (""), keeps the keys for the same trial in both registers in the returned vector.

prefermemberstate Code of single EU Member State for which records should returned. If not available, a record for GB or lacking this, any other record for the trial will be returned. For a list of codes of EU Member States, please see vector `countriesEUCTR`. Alternatively, "3RD" will lead to return a Third Country record of a trial, if available.

include3rdcountrytrials A logical value if trials should be retained that are conducted exclusively in third countries, that is, outside the European Union.

con A [src](#) connection object, as obtained with `nodbi::src_mongo()` or `nodbi::src_sqlite()`

verbose If set to TRUE, prints out information about numbers of records found at subsequent steps when searching for duplicates

Value

A vector with strings of keys ("_id" in the database) that represent non-duplicate trials.

Examples

```
## Not run:
db <- nodbi::src_sqlite(
  collection = "my_collection"
```



```

)
dbFindIdsUniqueTrials(
  con = db
)

## End(Not run)

```

dbGetFieldsIntoDf	<i>Create data frame by extracting specified fields from database collection</i>
-------------------	--

Description

With this convenience function, fields in the mongo database are retrieved into an R dataframe. As mongo json fields within the record of a trial can be hierarchical and structured, the function flattens the data and returns a concatenation of values if there is more than one value or if the field is (in) an array, such as follows: value 1 / value 2 / ... (see example)

Usage

```
dbGetFieldsIntoDf(fields = "", con, verbose = FALSE, stopifnodata = TRUE)
```

Arguments

fields	Vector of one or more strings, with names of the sought fields. See function dbFindFields for how to find names of fields.
con	A src connection object, as obtained with <code>nodbi::src_mongo()</code> or <code>nodbi::src_sqlite()</code>
verbose	Printing additional information if set to TRUE; default is FALSE.
stopifnodata	Stops with an error (TRUE, default) 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.

Details

For more sophisticated data retrieval from the database, see vignette examples and other packages to query mongodb such as `mongolite`.

Value

A data frame with columns corresponding to the sought fields. Note: a column for the record `_id` will always be included. The maximum number of rows of the returned data frame is equal to, or less than the number of records in the data base.

Examples

```
## Not run:
db <- nodbi::src_sqlite(
  collection = "my_collection"
)

# access fields that are nested within another field
# and can have multiple values with the other field
dbGetFieldsIntoDf(
  "b1_sponsor.b31_and_b32_status_of_the_sponsor",
  con = db
)[1,]
#           _id b1_sponsor.b31_and_b32_status_of_the_sponsor
# 1 2004-000015-25-GB           Non-commercial / Commercial

# access fields that include a list of values
# which are printed as comma separated values
dbGetFieldsIntoDf(
  "keyword",
  con = db
)[1,]

#           _id           keyword
# 1 NCT00129259 T1D, type 1 diabetes, juvenile diabetes

str(.Last.value)
# 'data.frame': 1 obs. of  2 variables:
# $ _id      : chr "NCT00129259"
# $ keyword:List of 1
# ..$ : chr "T1D" "type 1 diabetes" "juvenile diabetes"

## End(Not run)
```

dbQueryHistory
Show the history of queries that were loaded into a database

Description

Show the history of queries that were loaded into a database

Usage

```
dbQueryHistory(con, verbose = FALSE)
```

Arguments

con A [src](#) connection object, as obtained with `nodbi::src_mongo()` or `nodbi::src_sqlite()`
verbose If TRUE, prints additional information (default FALSE).

Value

A data frame 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, and with one row for each [ctrLoadQueryIntoDb](#) loading trial records in this collection.

Examples

```
## Not run:
db <- nodbi::src_sqlite(
  collection = "my_collection"
)
dbQueryHistory(
  con = db
)

## End(Not run)
```

dfListExtractKey

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

Description

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")))
```

Arguments

df	A data frame
list.key	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.

Value

A data frame 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

```
## Not run:
db <- nodbi::src_sqlite(
  collection = "my_collection"
)
df <- dbGetFieldsIntoDf(
  fields = c(
    "endPoints.endPoint",
    "subjectDisposition.postAssignmentPeriods"),
  con = db
)
dfListExtractKey(
  df = df,
  list.key = list(
    c("endPoints.endPoint",
      "^title"),
    c("subjectDisposition.postAssignmentPeriods",
      "arms.arm.type.value")
  )
)
## End(Not run)
```

dfMergeTwoVariablesRelevel

Merge two variables into one, optionally map values to new levels

Description

Merge two variables into one, optionally map values to new levels

Usage

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

Arguments

df	A data.frame in which there are two variables (columns) to be merged into one.
colnames	A vector of length two with names of the two columns that hold the variables to be merged. See colnames for how to obtain the names of columns of a data frame.
levelslist	A list with one slice each for a new value to be used for a vector of old values (optional).
...	for deprecated varnames parameter (will be removed)

Value

A vector of strings

Examples

```
## Not run:
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 = result,
  colnames = c("Recruitment", "x5_trial_status"),
  levelslist = statusvalues)

## End(Not run)
```

dfName2Value	<i>Extract information of interest (e.g., endpoint) from long data frame of protocol- or result-related trial information as returned from dfTrials2Long</i>
--------------	--

Description

Extract information of interest (e.g., endpoint) from long data frame of protocol- or result-related trial information as returned from [dfTrials2Long](#)

Usage

```
dfName2Value(df, valuenname = "", wherenname = "", wherevalue = "")
```

Arguments

df	A data frame with 5 columns (trial_id, main_id, sub_id, name, value) as returned from dfTrials2Long
valuenname	A character string for the name of the variable from which to extract information for the variable of interest
wherenname	A character string to identify the variable of interest
wherevalue	A character string with the value of interest for the variable of interest

Value

A data frame with columns trial_id, main_id, sub_id, name, value that only includes the values of interest

Examples

```

## Not run:
db <- nodbi::src_sqlite(
  collection = "my_collection"
)
df <- ctrdata::dbGetFieldsIntoDf(
  fields = c(
    # ctgov - typical results fields
    "clinical_results.baseline.analyzed_list.analyzed.count_list.count",
    "clinical_results.baseline.group_list.group",
    "clinical_results.baseline.analyzed_list.analyzed.units",
    "clinical_results.outcome_list.outcome",
    "study_design_info.allocation",
    # euctr - typical results fields
    "trialInformation.fullTitle",
    "subjectDisposition.recruitmentDetails",
    "baselineCharacteristics.baselineReportingGroups.baselineReportingGroup",
    "endPoints.endPoint",
    "trialChanges.hasGlobalInterruptions",
    "subjectAnalysisSets",
    "adverseEvents.seriousAdverseEvents.seriousAdverseEvent"
  ), con = dbc
)
# convert to long
reslong <- ctrdata::dfTrials2Long(
  df = df
)
# get values for endpoint of interest, duration of response
ctrdata::dfValue2Name(
  df = df,
  valuname = paste0(
    "endPoints.endPoint.*armReportingGroup.tendencyValues.tendencyValue.value|",
    "clinical_results.*category.measurement_list.measurement.value|",
    "clinical_results.*outcome.measure.units|endPoints.endPoint.unit"
  ),
  wherename = "clinical_results.*outcome.measure.title|endPoints.endPoint.title",
  wherevalue = "duration of response"
)

## End(Not run)

```

dfTrials2Long

Extract trial information into long format

Description

The function works with protocol- and results- related information. It converts lists and other values into individual rows of a long data frame. From the resulting data frame, values of interest can then be selected (e.g. select and outcome and its analysis by number of the measure which has "Hazard Ratio" in its name).

Usage

```
dfTrials2Long(df)
```

Arguments

df Data frame with columns including the trial identifier (`_id`) and one or more variables as obtained from [dbGetFieldsIntoDf](#)

Value

A data frame with the five columns: `trial_id`, `main_id`, `sub_id`, `name`, `value`

Examples

```
## Not run:
db <- nodbi::src_sqlite(
  collection = "my_collection"
)
df <- dbGetFieldsIntoDf(
  fields = c(
    "clinical_results.outcome_list.outcome"),
  con = db
)
dfTrials2Long(
  df = df
)

## End(Not run)
```

```
installCygwinWindowsDoInstall
```

Convenience function to install a minimal cygwin environment under MS Windows, including perl, sed and php

Description

Alternatively and in case of difficulties, download and run the cygwin setup yourself as follows:
`cygwinsetup.exe --no-admin --quiet-mode --verbose --upgrade-also --root c:/cygwin --site http://www.mirror-service.org/sites/sourceware.org/pub/cygwin/ --packages perl,php-jsonc,php-simplexm`

Usage

```
installCygwinWindowsDoInstall(force = FALSE, proxy = "")
```

Arguments

force	Set to TRUE to force updating and overwriting an existing installation in c:\cygwin
proxy	Specify any proxy to be used for downloading via http, e.g. "host_or_ip:port". installCygwinWindowsDoInstall may detect and use the proxy configuration used in MS Windows to use an automatic proxy configuration script. Authenticated proxies are not supported at this time.

Index

colnames, [12](#)
ctrdata, [4](#)
ctrFindActiveSubstanceSynonyms, [2](#)
ctrGetQueryUrlFromBrowser, [3](#), [4](#), [6](#)
ctrLoadQueryIntoDb, [3](#), [4](#), [11](#)
ctrOpenSearchPagesInBrowser, [3](#), [6](#)

data.frame, [12](#)
dbFindFields, [7](#), [9](#)
dbFindIdsUniqueTrials, [8](#)
dbGetFieldsIntoDf, [9](#), [11](#), [15](#)
dbQueryHistory, [3](#), [4](#), [6](#), [10](#)
dfListExtractKey, [11](#)
dfMergeTwoVariablesRelevel, [12](#)
dfName2Value, [13](#)
dfTrials2Long, [13](#), [14](#)

installCygwinWindowsDoInstall, [15](#)

src, [5](#), [7–10](#)
src_mongo, [5](#), [7–10](#)
src_sqlite, [5](#), [7–10](#)