Examples for using R package `ctrdata` for clinical trial information

Ralf Herold

2019-09-29

Contents

```r
# helper functions
#
normalise_number <- function(x) {

  x <- gsub("&lt;|&gt;", "",  x)
  x <- gsub("<=|<| |=|>", "",  x)
  x <- gsub("(^| )[.]", "0.", x)
  x <- trimws(x)
  x <- sapply(x, function(y) {
    z <- try(as.numeric(y), silent = TRUE)
```

```r
    if (class(z) == "try-error") {
      NA
      message(z)
    } else {
      z
    }
  })
  unname(x)
}
#
normalise_string <- function(x) {

  x <- gsub(",", "",  x)
  x <- gsub("-", " ", x)
  x <- tolower(x)
  x <- tools::toTitleCase(x)
  x <- gsub("[ ]+", " ", x)
  x <- trimws(x)
  x

}
#
grepl_multi <- function(patterns, x, simplify = TRUE) {
  if (!is.vector(patterns)) stop("patterns should be a vector.")
  ret <- apply(as.matrix(patterns), 1, function(pattern) grepl(pattern, x, ignore.case = TRUE))
  ret <- as.data.frame(ret)
  if (simplify)
    ret <- rowSums(ret) >= 1
  else
    names(ret) <- patterns
  return(ret)
}
#
age_ctgov <- function(x) {

  tmp.years  <- suppressWarnings(as.numeric(gsub("([0-9]+) Year.?",  "\\1", x, ignore.case = TRUE))) /
  tmp.months <- suppressWarnings(as.numeric(gsub("([0-9]+) Month.?", "\\1", x, ignore.case = TRUE))) /
  tmp.days   <- suppressWarnings(as.numeric(gsub("([0-9]+) Day.?",   "\\1", x, ignore.case = TRUE))) /

  tmp.age <- ifelse(!is.na(tmp.years),  tmp.years,
                ifelse(!is.na(tmp.months), tmp.months,
                    ifelse(!is.na(tmp.days),   tmp.days, NA)))

  return(tmp.age)
}
#
getKeyNestedList <- function(thelist, thekey) {

  # recursive find
  rmatch <- function(x, name) {

    pos <- match(name, names(x))
    if (!is.na(pos)) return(x[[pos]])
```

```
    for (element in x) {
      if (any(c("data.frame", "list") %in% class(element))) {
        out <- Recall(element, name)
        if (!is.null(out)) return(out)
      }
    }
  }

  # get key from any level, deduplicate, remove NA,
  # collapse if more than one value per list item
  sapply(thelist, function(x) {
    out <- rmatch(x = x, name = thekey)
    out <- na.omit(out)
    out <- unique(out)
    out <- paste0(out, collapse = " / ")
    out
  })
}
#
simplifyList <- function(l) {

  # helper function to avoid
  # that NULL is ignored / removed
  rapply2 <- function(x, fn) {
    if (is.list(x)) {
      sapply(x, rapply2, fn)
    } else {
      fn(x)
    }
  }

  # apply function replacing
  # NULL with e.g. NA
  tmp <- rapply2(l,
                 function(x) if (is.null(x)) NA else x)

  # recursively unlist and change to vector
  sapply(tmp, function(x) as.vector(unlist(x)))

}
```

General information on the **ctrdata** package is available here: https://github.com/rfhb/ctrdata.

0.1 Internet access via proxy

Functions in package **ctrdata** that start with `ctr...` require access to internet resources via `https`. Package **ctrdata** checks and automatically uses the proxy that is set under MS Windows in system settings. However, proxy settings need to be set by the user for other operating systems and for authenticating proxies, such as follows:

```
Sys.setenv(https_proxy = "your_proxy.server.domain:8080")
Sys.setenv(https_proxy_user = "userid:password")
```

0.2 Connect to a database

Here using MongoDB, which is faster than SQLite, can handle credentials, provides access to remote servers and can directly retrieve nested elements from paths. See README.md and Get started with ctrdata for examples using SQLite.

```
db <- nodbi::src_mongo(url = "mongodb://localhost",
                       db = "some_database_name",
                       collection = "some_collection_name")
db
# MongoDB 3.6.8 (uptime: 244492s)
# URL: laptop.home/some_database_name
# Collection: some_collection_name
```

0.3 Execute a search query and load into database

```
# Load library which has previously been installed:
library(ctrdata)

# In the browser, this is a search query used as example:
# all phase 3 adult breast cancer trials
q <- "https://www.clinicaltrialsregister.eu/ctr-search/search?query=cancer+AND+breast&age=adult&phase=pl

# Load details on all clinical trials into database:
ctrLoadQueryIntoDb(queryterm = q,
                   con = db)
# = Imported or updated 1896 records on 398 trial(s).
# * Updated history in meta-info of "some_collection_name"
```

0.4 Execute in browser a search query that was previously stored

The function `ctrOpenSearchPagesInBrowser()` can be used to open blank search pages, but also to execute a previous search. This can be used to check if user searches are properly handled by this package, for example by comparing the number of search results.

```
# After records have already been downloaded:
q <- dbQueryHistory(con = db)      # list the searches in the database (a data frame)
q <- q[2,]                         # select exactly one of the searches (by subsetting q)
ctrOpenSearchPagesInBrowser(q)     # open selected search newly in broweser

# This opens a browser and executes a search:
ctrOpenSearchPagesInBrowser("https://www.clinicaltrialsregister.eu/ctr-search/search?query=cancer")
```

0.5 Find fields / variables of interest

The search for fields is cached and thus accelerated during the R session and as long as no new ctrLoadQueryIntoDb() is executed.

```
dbFindFields(namepart = "date",
             con = db)
#  [1] "firstreceived_date"
#  [2] "lastchanged_date"
#  [3] "required_header.download_date"
#  [4] "verification_date"
#  [5] "start_date"
#  [6] "primary_completion_date"
#  [7] "completion_date"
#  [8] "x6_date_on_which_this_record_was_first_entered_in_the_eudract_database"
#  [9] "n_date_of_competent_authority_decision"
# [10] "n_date_of_ethics_committee_opinion"
# [11] "firstreceived_results_date"
# [12] "p_date_of_the_global_end_of_the_trial"
```

0.6 ctrdata returns include metadata

The objects returned by functions of this package include attributes with metadata to indicate from which query, database, collection etc. and when the object was created. Metadata can subsequently be reused.

```
attributes(dbGetFieldsIntoDf("e52_secondary_end_points",
                             con = db))
# $class
# [1] "data.frame"
#
# $`ctrdata-dbname`
# [1] "some_database_name"
#
# $`ctrdata-table`
# [1] "some_collection_name"
#
# $`ctrdata-dbqueryhistory`
#      query-timestamp query-register query-records
# 1 2019-08-11 16:54:47         EUCTR           444
# 2 2019-08-11 17:16:56         EUCTR           444
# 3 2019-08-12 20:05:27         EUCTR          1896
#                                        query-term
# 1          query=cancer&age=under-18&phase=phase-one
# 2          query=cancer&age=under-18&phase=phase-one
# 3 query=cancer+AND+breast&age=adult&phase=phase-three
```

0.7 Update data base from query in specified register

```
# List queries:
dbQueryHistory(con = db)

# Update last query in the history:
#  - EUCTR information can be incrementally downloaded or updated only
#    for the last 7 days, after which a full new download is necessary.
#  - CTGOV intformation can always be incrementally downloaded or updated.
```

```
#  - Alternatively, querytoupdate can have an integer number,
#    corresponding to the desired row of dbQueryHistory().
ctrLoadQueryIntoDb(querytoupdate = "last",
                   con = db)

dbQueryHistory(con = db)
# Note: after such an update, the column "query-records" refers
# to the number of records that were added or updated, only,
# when the update was run as per the "query-timestamp",
# not to the total number of records.

# The date and time of the last download of individual records
# is stored in the variable "record_last_import" as follows:
resultDate <- dbGetFieldsIntoDf("record_last_import",
                                con = db)

summary(resultDate[["record_last_import"]])
#                  Min.             1st Qu.               Median   [...]
# "2016-04-21 19:49:54" "2016-04-21 19:49:54" "2016-04-21 19:49:54"  [...]
```

0.8   Deduplicate country-specific records of a trial, visualise trial information

Note EUCTR provides country-specific records for trials, see README.md.

```
# Example 1: Relation between number of study participants in one country and in whole trial?
result <- dbGetFieldsIntoDf(c("f41_in_the_member_state",
                              "f422_in_the_whole_clinical_trial"),
                            con = db)

plot(f41_in_the_member_state ~ f422_in_the_whole_clinical_trial, result)
# [result not shown]

# Example 2: How many clinical trials are ongoing or completed, per country?
result <- dbGetFieldsIntoDf(c("a1_member_state_concerned",
                              "p_end_of_trial_status"),
                            con = db)

with(result,
     table(a1_member_state_concerned,
           p_end_of_trial_status))
# [result not shown]

# Example 3: How many clinical trials where started in which year?
result <- dbGetFieldsIntoDf(c("a1_member_state_concerned",
                              "n_date_of_competent_authority_decision",
                              "a2_eudract_number"),
                            con = db)

# Eliminate trials records duplicated by member state:
result <- result[ result[["_id"]] %in%
                  dbFindIdsUniqueTrials(con = db), ]
```

```
# Visualise:
hist(result[["n_date_of_competent_authority_decision"]],
     breaks = "years",
     freq = TRUE,
     las = 1); box()
#
```

0.9 Download from another register, check for duplicates, merge variables and re-organise values

```
# Ongoing phase 3 interventional trials in breast carcinoma
q <- paste0("https://clinicaltrials.gov/ct2/results?term=&recr=Open&",
            "type=Intr&cond=breast+carcinoma&intr=Drug&state1=&age=1&phase=2")

# Show and check search in browser:
ctrOpenSearchPagesInBrowser(q)

# Count number of trials:
ctrLoadQueryIntoDb(q,
                   only.count = TRUE,
                   con = db)

# Load data:
ctrLoadQueryIntoDb(q,
                   con = db)

# Data from which queries have now been downloaded into the database?
dbQueryHistory(con = db)

# Get columns from the database from different registers
dbFindFields("status",
             con = db)

result <- dbGetFieldsIntoDf(c("overall_status",
                              "p_end_of_trial_status",
                              "a2_eudract_number"),
                            con = db)

# Find ids of unique trials if recorded by alternative countries or registers:
ids_of_unique_trials <- dbFindIdsUniqueTrials(prefermemberstate = "FR", con = db)
ids_of_unique_trials <- dbFindIdsUniqueTrials(preferregister = "EUCTR", con = db)
ids_of_unique_trials <- dbFindIdsUniqueTrials(preferregister = "CTGOV", con = db)
# * Total of 2488 records in collection.
# Searching for duplicates, found
#  - 1788 EUCTR _id were not preferred EU Member State record of trial
#  - 22 EUCTR _id in CTGOV secondary_id / nct_alias / org_study_id
#  - 12 EUCTR a52_us_nct_... in CTGOV _id (nct)
#  - 0 EUCTR a52_us_nct_... in CTOGV secondary_id / nct_alias / org_study_id
#  - 0 EUCTR a51_isrctn_... in CTOGV secondary_id / nct_alias / org_study_id
#  - 0 EUCTR a41_sponsors_protocol_... in CTOGV secondary_id / nct_alias / org_study_id
# Concatenating 146 records from CTGOV and 512 from EUCTR:
```

Figure 1: Histogram

```
# = Returning keys (_id) of 658 out of total 2488 records in collection "some_collection_name".

# Subset the result set to these unique trials:
result <- result[ result[["_id"]] %in% ids_of_unique_trials, ]

# Now merge two variables into a new variable for analysis:
tmp <- dfMergeTwoVariablesRelevel(result,
                                  colnames = c("overall_status",
                                               "p_end_of_trial_status"))
# Unique values returned: Ongoing, Completed, Prematurely Ended, Temporarily Halted,
#  Restarted, Recruiting, Not yet recruiting

table(tmp)
#                         Completed Not yet recruiting              Ongoing
#                43                154                 26                  265
# Prematurely Ended         Recruiting          Restarted Temporarily Halted
#                48                120                  1                    1

# Merge two variables as above and in addition,
# condense their values into a new value:
statusvalues <- list("ongoing" = c("Recruiting", "Active", "Ongoing",
                                    "Active, not recruiting",
                                    "Enrolling by invitation", "Restarted"),
                     "completed" = c("Completed", "Prematurely Ended", "Terminated"),
                     "other"     = c("Withdrawn", "Suspended", "No longer available",
                                     "Not yet recruiting", "Temporarily Halted",
                                     "Unknown status"))

tmp <- dfMergeTwoVariablesRelevel(result,
                                  colnames = c("overall_status",
                                               "p_end_of_trial_status"),
                                  levelslist = statusvalues)
# Unique values returned: ongoing, completed, other

table(tmp)
# tmp
#    completed    ongoing      other
#          159        671        115
```

0.10   Use a remote mongo database

This example works with a free service here. Note that the user name and password need to be encoded. The format of the connection string is documented here.

```
## Specify base uri for remote mongodb server as part of the encoded connection string
db <- nodbi::src_mongo(
  url = "mongodb+srv://DWbJ7Wh:bdTHh5cS@cluster0-b9wpw.mongodb.net",
  db = "dbperm",
  collection = "dbperm")

# Database connection can just be used
result <- dbGetFieldsIntoDf(
  fields = c("a2_eudract_number",
```

```
                  "overall_status",
                  "record_last_import",
                  "primary_completion_date",
                  "x6_date_on_which_this_record_was_first_entered_in_the_eudract_database",
                  "e71_human_pharmacology_phase_i"),
        con = db)
```

0.11 Use another package for analysis, for example count sites per trial with mongolite

Note: Packages to access mango such as `mongolite` or `RMongo` may work to access data that are already in the mongo dababase. However, package `ctrdata` is still necessary to initially retrieve data from registers and to store these data into the database.

```
library(mongolite)

m <- mongo(url = "mongodb+srv://DWbJ7Wh:bdTHh5cS@cluster0-b9wpw.mongodb.net/dbperm",
           collection = "dbperm")

# List all collections:
m$run('{"listCollections": 1}')

# Find the database elements that represent a site:
result <- m$find('{}', '{"location.facility.name": 1}')

# Table by frequency
tbl <- table(sort(unname(rapply(result, function(x) x, how = "unlist"))))
tbl <- tbl[rev(order(tbl))]
head(tbl)
#                       Children's Hospital of Philadelphia
#                                                        17
#              Cincinnati Children's Hospital Medical Center
#                                                        14
#                              Hospital for Sick Children
#                                                        13
#                        Children's National Medical Center
#                                                        13
# Children's Hospitals and Clinics of Minnesota - Minneapolis
#                                                        13
#                             Nationwide Children's Hospital
#                                                        11

# Define a helper function to count elements, irrespective of whether
# the elements are in an array or are a set of subdocuments in the database:
count.elements <- function(dataframecolumn) {
  return(sapply(dataframecolumn,
            function(x) ifelse (is.data.frame(tmp <- unlist (x[[1]])),
                                nrow(tmp), length(tmp))))
}

# Sum up number of sites per trial:
result$number_of_sites <- count.elements(result$location)
```

```
# Inspect first rows:
head(result[, c("_id", "number_of_sites")])
#                  _id number_of_sites
# 1       NCT01471782              30
# 2         meta-info               0
# 3 2010-024264-18-DE               0
# 4 2010-024264-18-AT               0
# 5 2010-024264-18-NL               0
# 6 2010-024264-18-IT               0

# For many trials, no locations seem to be specified:
result <- result[result$number_of_sites >= 1 &
                   result$number_of_sites < 20, ]

# Draw histogram:
hist(result$number_of_sites)
```

0.12   Working with nested data

Compare the way that fields (keys) may have to be specified differently for elements that are nested within other elements. See also Analyse results-related information below.

```
# with MongoDB: finds all elements b31_and_b32_...
result <- dbGetFieldsIntoDf(
  fields = c("b1_sponsor.b31_and_b32_status_of_the_sponsor", # Note dot path
            "p_end_of_trial_status",
            "a2_eudract_number"),
  con = db)

# with SQLite: finds the first, to retrieve
# the second, increment to [1] and so on
result <- dbGetFieldsIntoDf(
  fields = c("b1_sponsor[0].b31_and_b32_status_of_the_sponsor", # Note [0]
            "p_end_of_trial_status",
            "a2_eudract_number"),
  con = db)

# with SQLite (and MongoDB): retrieve element at root
# and extract its nested sub-elements later, using
# the helper function that follows below.
result <- dbGetFieldsIntoDf(
  fields = c("b1_sponsor",
            "p_end_of_trial_status",
            "a2_eudract_number"),
  con = db)

# Nested lists: its elements can be accessed
# using the helper function getKeyNestedList()
result[1, "b1_sponsor"]

# Example
getKeyNestedList(thelist = result$b1_sponsor[1:2],
```

```
                 thekey = "b11_name_of_sponsor")

# Eliminate trials records duplicated by EU Member State:
uniqueids <- dbFindIdsUniqueTrials(con = db)
result     <- result[ result[["_id"]] %in% uniqueids, ]

# Tabulate the status of the clinical trial on the date of information retrieval
# Note some trials have more than one sponsor and values are concatenated with /.
with(result, table(p_end_of_trial_status,
                   b1_sponsor.b31_and_b32_status_of_the_sponsor))
#                  b1_sponsor.b31_and_b32_status_of_the_sponsor
# p_end_of_trial_status    Commercial  Non-Commercial  Non-Commercial / Non-Commercial
#    Completed                     81              32                                0
#    Ongoing                      205             239                               12
#    Prematurely Ended             15              12                                0
#    Restarted                      0               1                                0
#    Temporarily Halted             4               1                                0
```

0.13 Plot frequency of certain end points

```
# Search for interesting variables,
# note the spelling in EUCTR:
dbFindFields("end_point",
             con = db)

# Get interesting variables from database
# for further analysis within R:
result <- dbGetFieldsIntoDf(c("a2_eudract_number",
                              "a41_sponsors_protocol_code_number",
                              "n_date_of_competent_authority_decision",
                              "e73_therapeutic_confirmatory_phase_iii",
                              "e51_primary_end_points"),
                            con = db)

# Eliminate trials records duplicated by member state
# keep a single record and if available use preference:
result <- result[ result[["_id"]] %in%
                  dbFindIdsUniqueTrials(con = db), ]

# Checking expected number of trials:
nrow(result)
length(unique(result$a2_eudract_number))

# Only use phase 3 trials:
table(result$e73_therapeutic_confirmatory_phase_iii, exclude = "")
result <- result[result$e73_therapeutic_confirmatory_phase_iii, ]

# Is the a primary endpoint of interest? This uses regular expressions:

# PFS / EFS / RFS / DFS
result$pe_is_efs <- grepl(
```

```
  pattern = "((progression|event|relapse|recurrence|disease)[- ]free)|pfs|dfs|efs)",
  x = result$e51_primary_end_points,
  ignore.case = TRUE)

# Plot:
library(ggplot2)
ggplot(data = result,
       aes(x = n_date_of_competent_authority_decision,
           fill = pe_is_efs)) +
  geom_histogram(binwidth = 365.25)  +
  labs(title = "Breast cancer phase 3 clinical trials",
       x = "Year of clinical trial authorisation in EU",
       y = "Number of clinical trials") +
  scale_fill_manual(name = "Primary endpoint",
                    values = c("red", "green"),
                    labels = c("FALSE" = "Other", "TRUE" = "PFS/EFS/DFS")) +
  theme(axis.text.x  = element_text(angle = 90, vjust = 0.5)) +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y")

# ggsave("vignettes/Rplot02.png", width = 6, height = 4)

# Plausibility check - what is the primary endpoint
# if not one of the endpoints of interest?
# Look into first ten examples foundL
substr(result$e51_primary_end_points[result$pe_is_efs == FALSE], 1, 60)[1:8]
# [results not shown]
```

0.14   Histogram of status of investigational medicinal product

```
if (FALSE) ctrLoadQueryIntoDb(
  register = "EUCTR",
  queryterm = "&age=under-18&phase=phase-one&phase=phase-two",
  con = db)

dbQueryHistory(con = db)

ctrOpenSearchPagesInBrowser(dbQueryHistory(con = db))

if (FALSE) ctrLoadQueryIntoDb(
  querytoupdate = "last",
  con = db)

dbFindFields(namepart = "phase",
             con = db)
# [1] "e72_therapeutic_exploratory_phase_ii"    "e71_human_pharmacology_phase_i"
# [3] "e73_therapeutic_confirmatory_phase_iii" "e74_therapeutic_use_phase_iv"

result <- dbGetFieldsIntoDf(
  c("a1_member_state_concerned", "n_date_of_competent_authority_decision",
    "dimp.d21_imp_to_be_used_in_the_trial_has_a_marketing_authorisation",
    "e11_medical_conditions_being_investigated", "e112_therapeutic_area",
```

Figure 2: Histogram2

```r
    "e71_human_pharmacology_phase_i", "e72_therapeutic_exploratory_phase_ii",
    "x6_date_on_which_this_record_was_first_entered_in_the_eudract_database",
    "f422_in_the_whole_clinical_trial",
    "a3_full_title_of_the_trial", "a2_eudract_number"),
  con = db)

searchfor <- paste0("cancer, leukaem, leukem, tumour, tumor, sarcoma, blastom, gliom, germ, ",
                    "lymphom, malign, hodgkin, ewing, rhabdo, terato, tumeur, leucemi")
searchfor <- strsplit(searchfor, ", ")[[1]]

relevant <- grepl_multi(searchfor, result$a3_full_title_of_the_trial) |
  grepl_multi(searchfor, result$e11_medical_conditions_being_investigated) |
  grepl_multi("C04",     result$e112_therapeutic_area)
table(relevant)

# get relevant trials
result <- result[relevant,]
result <- result[result$e71_human_pharmacology_phase_i,]

# find first member state to authorise trial
tmp <- aggregate(result$n_date_of_competent_authority_decision,
                 by = list(result$a2_eudract_number),
                 FUN = function(x) min(x))
result <- merge(x = result, y = tmp,
                by.x = "a2_eudract_number",
                by.y = "Group.1",
                all.x = TRUE)

# label date and calculate year
names(result)[ncol(result)] <- "startdatefirst"
result$startyearfirst <- as.numeric(format(result$startdatefirst, "%Y"))

# eliminate country duplicates
result <- result[ result[["_id"]] %in%
                    dbFindIdsUniqueTrials(
                      include3rdcountrytrials = FALSE,
                      con = db), ]

# any of the investigational medicinal product(s) authorised or not?
tmp <- stringi::stri_count(
  result$dimp.d21_imp_to_be_used_in_the_trial_has_a_marketing_authorisation,
  fixed = "Yes")
table(tmp, exclude = "")
tmp <- tmp > 0

# draw plot
library(ggplot2)
ggplot(data = result,
       aes(x = startyearfirst, fill = tmp)) +
  geom_histogram(binwidth = 1) +
  labs(title = "Paediatric phase 1 oncology trials in EU",
       x = "Year of trial authorisation (or EUCTR upload)",
       y = "Number of trials",
```

Figure 3: Histogram3

```
        fill = "Medicine\nauthorised?")

# save plot
# ggsave("vignettes/phase1_paed_oncol.png", width = 6, height = 4)
```

0.15 Plot map of ongoing trials

```
# this is to create a map of EU ongoing phase 1 trials with children with cancer on an anti-cancer medi
library(rworldmap)
library(plyr)

# get relevant variables from relevant collection
result <- dbGetFieldsIntoDf(
  c("a1_member_state_concerned", "n_date_of_competent_authority_decision",
    "e11_medical_conditions_being_investigated", "e112_therapeutic_area",
    "e71_human_pharmacology_phase_i", "e72_therapeutic_exploratory_phase_ii",
    "x6_date_on_which_this_record_was_first_entered_in_the_eudract_database",
    "a3_full_title_of_the_trial", "a2_eudract_number", "p_end_of_trial_status",
    "ctrname"),
  con = db)
```

```r
# calculate country variable
result$country <- sub("^.+-([3A-Z]+)$", "\\1", result$`_id`)

# select trials of interest
searchfor <- paste0("cancer, leukaem, leukem, tumour, tumor, sarcoma, blastom, gliom, germ, lymphom, ",
                    "malign, hodgkin, ewing, rhabdo, terato, tumeur, leucemi")
searchfor <- strsplit(searchfor, ", ")[[1]]

relevant <- grepl_multi(searchfor, result$a3_full_title_of_the_trial) |
  grepl_multi(searchfor, result$e11_medical_conditions_being_investigated) |
  grepl_multi("C04",     result$e112_therapeutic_area)

table(relevant)

# apply selection criteria
result <- result[relevant,]
result <- result[result$e71_human_pharmacology_phase_i,]
result <- result[result$ctrname == "EUCTR",]
result <- result[result$country != "3RD",]

# get an overview
counts <- as.data.frame.matrix(
  with(result,
       table(country, p_end_of_trial_status)))

counts$total <- rowSums(counts)
counts$country <- rownames(counts)
counts$text <- with(counts, paste(Ongoing, total, sep = "/"))
counts

# plot
png(filename = "vignettes/ph1paedoncopen.png",
    width = 6,
    height = 5,
    units = "in",
    res = 300)
sPDF <- joinCountryData2Map(dF = counts,
                            joinCode = "ISO2",
                            nameJoinColumn = "country")
mapParams <- mapCountryData(mapToPlot = sPDF,
                            nameColumnToPlot = "Ongoing",
                            mapTitle = "",
                            xlim = c(-10,30),
                            ylim = c(36, 65),
                            addLegend = FALSE,
                            colourPalette = "terrain")
# do.call(addMapLegend, c(mapParams, legendLabels="all", legendWidth=0.5, digits=0, labelFontSize=0.8))
title(main = "Number of ongoing / total phase 1\nanti-cancer medicine trials with children")
labelCountries(dF = sPDF,
               nameCountryColumn = "text",
               xlim = c(-15,32),
               ylim = c(36, 65),
               cex = 0.9,
```

Figure 4: Histogram3

```
                col = "blue")
dev.off()
```

0.16 Use aggregation functions of the data base to find specific trial endpoints

```
library(mongolite)

# Link R object m to the specified database collection:
m <- mongo(url = "mongodb://localhost/users",
           collection = "ctrdata")

# List all collections:
m$run('{"listCollections": 1}')

# Number of all records in this collection:
```

```
m$count()

# Number of EUCTR records, using json for query:
m$count(query = '{"_id": {"$regex": "[0-9]{4}-[0-9]{6}-[0-9]{2}", "$options": "i"}}')
# alternatively:
m$count(query = '{"ctrname": "EUCTR"}')

# Number of CTGOV records, using json for query:
m$count(query = '{"_id": {"$regex": "NCT[0-9]{8}", "$options": "i"}}')
# alternatively:
m$count(query = '{"ctrname": "CTGOV"}')

# The following uses the aggregation pipeline in mongo:
#
# Count number of records in which certain terms occur,
# in any of the elements of the array in primary_outcome.
# Regular expressions ("$regex") are case insensitive ("i")
#
# See here for details on mongo's aggregation pipleline:
# https://docs.mongodb.org/manual/core/aggregation-pipeline/

# Recommendation: to best define regular expressions for analyses,
# inspect the field primary_outcome.measure in database, or print:
m$distinct(key = "primary_outcome.measure",
           query = '{"_id": {"$regex": "NCT[0-9]{8}", "$options": "i"}}')[1:8]

# PFS, EFS, RFS or DFS
out <- m$aggregate(pipeline = '[{"$match": {"primary_outcome.measure":
                                {"$regex": "(progression|event|relapse|recurrence|disease)[- ]free",
                                           "$options": "i"}}},
                                {"$group": {"_id": "null", "count": {"$sum": 1}}}]')
out

# OS by year
out <- m$aggregate(pipeline = '[{"$match": {"primary_outcome.measure":
                                {"$regex": "overall survival",
                                           "$options": "i"}}},
                                {"$project": {"_id": 1, "start_date": 1}}]')

out
#             _id firstreceived_date
# 1   NCT00001941    January 18, 2000
# 2   NCT00002474    November 1, 1999
# 3   NCT00002495    November 1, 1999
# 4   NCT00002514    November 1, 1999
# [...]

# Extraction of year from firstreceived_date such as "August 29, 2009"
out$year <- substr(out$start_date,
                   tmp <- nchar(out$start_date) - 4,
                   tmp + 4)
#
table(out$year)
```

```
# [...]  2006  2007  2008  2009  2010  2011  2012  2013  2014  2015
#           5     4    13     3     8     3     8     4     6     6
#
```

0.17 Use mapreduce operations to analyse participant numbers

Note that the mongodb documentation includes that a mapreduce operation might not be as fast and efficient as using the aggregation pipeline, which was used in the preceding example.

```
library(mongolite)

m <- mongo(url = "mongodb://localhost/users",
           collection = "ctrdata")

# Count number of trials (trial records) with number of
# study participants in bins of hundreds of participants:
hist <- m$mapreduce(
  map = "function(){emit(Math.floor(this.f422_in_the_whole_clinical_trial/100)*100, 1)}",
  reduce = "function(id, counts){return Array.sum(counts)}"
)

hist
#      _id value
# 1    NaN  5364
# 2      0   344
# 3    100   183
# 4    200    79
# 5    300   102
# 6    400    78

# Note: _id is the bin of hundreds of study participants,
# value is the number of studies in respective bin
plot(hist, type = "h", las = 1, xlim = c(0, 2000), ylim = c(0, 500),
     xlab = "Number stubjects", y = "Number of trials")
```

0.18 Analyse inclusion criteria

```
# Search for interesting variables,
# note the spelling in EUCTR:
dbFindFields(namepart = "crit",
             con = db)

# Get interesting variables from database
# for further analysis within R:
result <- dbGetFieldsIntoDf(
  c("e3_principal_inclusion_criteria",
    "eligibility.criteria.textblock"),
  con = db)

# Eliminate trials records duplicated by member state
```

Figure 5: Histogram2

```
# keep a single record and if available use preference:
result <- result[ result[["_id"]] %in% dbFindIdsUniqueTrials(con = db), ]

result <- dfMergeTwoVariablesRelevel(
  df = result,
  colnames = c("e3_principal_inclusion_criteria",
               "eligibility.criteria.textblock"))

# search for interesting terms
terms <- c("positive", "marker", "select")

table(grepl_multi(terms, result))

# utility function to generate a regular expression for a given number of words
words <- function(x) paste0(paste0(rep("\\s+\\w+", x), collapse = ""), "\\s+")

terms <- paste0(".*(", words(2), terms, "\\w*", words(3), ").*")

# find and print found matches for review
for (i in 1:length(terms))
  print(gsub(terms[i], "\\1",
             result,
             ignore.case = TRUE)[grepl(terms[i], result)])
```

0.19   Find trial start date

```
# Get interesting variables from database
# for further analysis within R:
result <- dbGetFieldsIntoDf(
  c("a2_eudract_number",
    "n_date_of_competent_authority_decision"),
  con = db)

# Calculate first date of trial start
first_trial_start <- aggregate(x = result$n_date_of_competent_authority_decision,
                               by = list(result$a2_eudract_number),
                               FUN = function(x) min(x, na.rm = TRUE),
                               simplify = TRUE, drop = TRUE)

# Give data frame appropriate names:
names(first_trial_start) <- c("a2_eudract_number",
                              "first_trial_start_date")

# Merge back into result set:
result <- merge(result, first_trial_start, all.x = TRUE)
```

0.20   Annotate records when using ctrLoadIntoDb() with user text

```r
if (FALSE) ctrLoadQueryIntoDb(
  queryterm = paste0("https://clinicaltrials.gov/ct2/results?age=0&intr=dasatinib&phase=1"),
  annotation.text = paste0("|AS:", "dasatinib", "|"),
  annotation.mode = "prepend",
  con = db)

# helper function
getannotationelement <- function(x, element = "AS") {

  # example from annotation.text:
  # x <- "|AS:tisagenlecleucel| |AS:dasatinib|"

  x <- strsplit(x = x, split = "\\|")
  x <- unlist(x)
  x <- x[grepl(paste0("^", element, ":"), x)]
  x <- strsplit(x = x, split = ":")
  x <- unlist(x)
  x <- x[!grepl("AS", x)]
  x <- sort(unique(x))

  return(x)
}

result <- dbGetFieldsIntoDf(
  fields = "annotation",
  con = db)

getannotationelement(
  result$annotation,
  element = "AS")

# [1] "dasatinib"
```

0.21   Analyse results-related information

Distribution of p values

```r
## get trials into database
if (FALSE) ctrLoadQueryIntoDb(
  queryterm = paste0("https://clinicaltrials.gov/ct2/results?rslt=With&age=0&intr=Drug&",
                     "phase=1&phase=2&strd_s=01%2F01%2F2010&strd_e=12%2F31%2F2012"),
  con = db)

## get result set
# dbFindFields(namepart = "age", allmatches = TRUE, collection = "resultsTest")
result <- dbGetFieldsIntoDf(
  c("start_date",
    "detailed_description.textblock",
    "official_title",
    "eligibility.maximum_age",
    "study_design_info.allocation",
    "primary_outcome.measure",
```

```r
    "arm_group.arm_group_type",
    "clinical_results.outcome_list.outcome.analysis_list.analysis.method",
    "clinical_results.outcome_list.outcome.analysis_list.analysis.p_value", # column 10
    "clinical_results.outcome_list.outcome.analysis_list.analysis.param_type",
    "clinical_results.outcome_list.outcome.analysis_list.analysis.non_inferiority_type",
    "clinical_results.baseline.analyzed_list.analyzed.units",
    "clinical_results.baseline.analyzed_list.analyzed.count_list.count.@attributes.value"
  ),
  con = db,
  stopifnodata = FALSE)


## count number of participants per trial
tmp <- simplifyList(result[, "clinical_results.baseline.analyzed_list.analyzed.count_list.count.@attribu

# change into number and sum up per trial
tmp <- sapply(tmp, function(x) sum(as.numeric(x), na.rm = TRUE))

# check that unit of first analysis is participants, not e.g. wounds or eyes
tmpparticipants <- sapply(result$clinical_results.baseline.analyzed_list.analyzed.units,
                          function(x) (x[[1]] == "Participants")[1])


## p value for endpoint analysis
tmp <- simplifyList(result[, "clinical_results.outcome_list.outcome.analysis_list.analysis.p_value"])

# get first p-value (first analysis primary endpoint)
tmp <- sapply(tmp, "[[", 1)

# replace empty list elements with NA
tmp <- sapply(tmp, function(x) ifelse(length(x), x, NA))

# turn strings into numbers, see helper function
tmp <- sapply(tmp, FUN = function(x) normalise_number(x))

summary(tmp)

result$pvalueprimaryanalysis <- tmp


## statistical method used for primary endpoint analysis
tmp <- simplifyList(result["clinical_results.outcome_list.outcome.analysis_list.analysis.method"])

# get first method (first analysis primary endpoint)
tmp <- sapply(tmp, "[[", 1)

# replace empty list elements with NA
tmp <- sapply(tmp, function(x) ifelse(length(x), x, NA))

table(tmp)

result$methodprimaryanalysis <- tmp
```

```r
## Based on allocation arms, keep trials that are likely to investigate safety and efficacy
## considering that active comparators may not be designed to show superiority.
## Note this checks the string with a regular expression and does not convert to a list:
## "list(arm_group_type = c(\"Active Comparator\", \"Placebo Comparator\", \"Experimental\"))"
# controlled?
result$control <- grepl("(Placebo|No Intervention)",
                        result$arm_group.arm_group_type,
                        ignore.case = TRUE)
table(result$control)
# randomised?
result$randomised <- grepl("^Randomi.*",
                           result$study_design_info.allocation,
                           ignore.case = TRUE)
table(result$randomised)
# two or more arms?
tmp <- simplifyList(result["arm_group.arm_group_type"])
tmp <- sapply(tmp, function(x) length(unlist(x)) >= 2)
result$twoormoregroups <- tmp
result <- subset(result, control & randomised & twoormoregroups)


## plot p values
# http://varianceexplained.org/statistics/interpreting-pvalue-histogram/
# http://www.pnas.org/content/100/16/9440.full
library(ggplot2)
ggplot(result, aes(pvalueprimaryanalysis)) +
  labs(title = paste0("Paediatric phase 2 or 3 interventional trials\n",
                      "with randomisation to placebo or to no intervention"),
      x = "p value in primary endpoint primary analysis",
      y = "Empirical cumulative density of trials") +
  geom_vline(aes(xintercept = 0.05,
                 colour = "red"),
             show.legend = FALSE) +
  stat_ecdf(geom = "step")
# ggsave("vignettes/ctrdatapvaluesphase23.png", width = 6, height = 4)

## statistical method used for primary endpoint analysis
tmp <- table(result$methodprimaryanalysis)
tmp <- tmp[rev(order(tmp))]
tmp <- data.frame(tmp)
knitr::kable(tmp[1:10,])
```

Var1	Freq
ANCOVA	41
Cochran-Mantel-Haenszel	14
Mixed Models Analysis	12
ANOVA	11
Fisher Exact	7
t-test, 2 sided	6
Chi-squared	6
Regression, Cox	3
Wilcoxon (Mann-Whitney)	2

Var1	Freq
Regression, Logistic	2

Figure 6: ECDFpvalues