Capr-PHEMS example usage

Guus @TheHyve

2024-08-29

Set-up knitr

Set-up config

```
connectionConfig <- config::get(config = 'config', file = './inst/config/connection_config.yml')
config_oth <- config::get(config = 'config', file = './inst/config/config.yml')
```

Load libraries

```
library(RSQLite)
library(tibble)
library(DatabaseConnector)
library(CohortGenerator)
```

```
## Loading required package: R6
```

```
library(CirceR)
library(Capr)
```

Create connection details

Here we specify our connection details. In this case, we use a yml configuration file to specify our connection details. Alternatively, strings may be used directly to assign the function arguments. These details will allow us to connect to the DB / OMOP CDM instance.

```
#  Use connection details from configuration
connectionDetails <- createConnectionDetails(
  dbms = connectionConfig$dbms,
  user = connectionConfig$user,
  password = connectionConfig$password,
  server = connectionConfig$server,
  port = connectionConfig$port,
  oracleDriver = connectionConfig$oracleDriver,
  pathToDriver = connectionConfig$pathToDriver
)
```

Concept sets

Concept sets have been defined per use case in ./R/conceptSets.R. Here we collect the details for all concept ids in all created concept sets using the Capr::getConceptSetDetails() function. This can only be done for concepts that exist within the concept set; not for descendants of these concepts. This function can be applied to any concept set and returns the concept_name, standardness, the domain, the source vocabulary and code, and more.

```
## Concept sets
source("./R/conceptSets.R")
```

```
##
## Sourced concept sets
```

```
# Establish connection
con <- connect(connectionDetails)
```

```
## Connecting using PostgreSQL driver
```

```
conceptSets$conceptSets <- conceptSets$conceptSets %>%
  # Add details for all concepts (excl. descendants)
  lapply(FUN = getConceptSetDetails,
         con = con,
         vocabularyDatabaseSchema = connectionConfig$vocabulary_schema)
```

```
## Warning: Input SQL has already been translated, so not translating again
## This warning is displayed once every 8 hours.
```

```
# Disconnect
disconnect(con)
```

Concept counts

Retrieve counts for all concepts in a concept set. Return the counts on the person (1) and record (2) level for each concept expli- citly in the concept set and on the person (3) and record (4) level including all descendants of concepts in the conceptSet.

```
## Count occurrences of each concept in data
# Establish connection
con <- connect(connectionDetails)
```

```
## Connecting using PostgreSQL driver
```

```
# Get countOccurrences function
source("./R/countOccurrences.R")

# Get links between tables and fields as input
source("./R/table_linked_to_concept_field.R")
```

```
## Sourced links object
```

```
uc1Counts <-
  countOccurrences(
    v = conceptSets$concepts$uc1,
    tables = names(links), # Query all CDM tables
    links = links, # Links between tables and concept_id fields (cdm_table:field)
    db_connection = con,
    cdm_schema = connectionConfig$cdm_schema,
    vocab_schema = connectionConfig$vocabulary_schema,
    save_path = config_oth$save_path_counts
  ) %>% print()
```

```
##   |                                                                      |
```

```
## Executing SQL took 0.0155 secs
```

```
##   |                                                                      |
```

```
## Executing SQL took 0.00728 secs
## `summarise()` has grouped output by 'concept_id', 'concept_name'. You can override using the `.groups
```

```
## # A tibble: 198 x 7
##    concept_id concept_name                 domain_id count_persons count_records
##         <dbl> <chr>                        <chr>             <dbl>         <dbl>
##  1    3004249 Systolic blood pressure      Measurem~          1002         13250
##  2    3012888 Diastolic blood pressure     Measurem~          1002         13250
##  3    3025315 Body weight                  Measurem~          1002         13250
##  4    3036277 Body height                  Measurem~          1002         13094
##  5    3024171 Respiratory rate             Measurem~          1002         13054
##  6    3027018 Heart rate                   Measurem~          1002         13054
##  7    1343916 epinephrine                  Drug                  0             0
##  8    4020553 Oxygen saturation measureme~ Measurem~             0             0
##  9    4301351 Surgical procedure           Procedure             0             0
## 10    4163872 Plain chest X-ray            Procedure            46            47
## # i 188 more rows
## # i 2 more variables: desc_count_person <dbl>, desc_count_record <dbl>
```

```
# Disconnect
disconnect(con)
```

Standard and non-standard concepts given a database connection

The isStandardDB function checks for every concept used in a CDM instance whether it is standard or not. It works by joining the CDM tables to the concept table in the vocabulary schema on the concept_id. The function returns a table of non-standard concepts.

```
# Connect to DB
con <- connect(connectionDetails)
```

```
## Connecting using PostgreSQL driver
```

```r
# Return table of non-standard concepts
source('./R/isStandardDB.R')
nonStandardDB <- isStandardDB(
  db_connection = con,
  cdm_schema = connectionConfig$cdm_schema,
  vocab_schema = connectionConfig$vocabulary_schema,
  links = links, # Links between tables and concept_id fields (cdm_table:field)
  save_path = config_oth$save_path_isStandard
) %>% print()
```

```
## No non-standard concepts found in DB; not saving any file.
##
## Finished checking for non-standard concepts.
## 0 non-standard concepts found across tables.
```

```
## [1] concept_id       concept_name     source_code      standard_concept
## [5] source_table
## <0 rows> (or 0-length row.names)
```

Standard and non-standard concepts given a list of concept IDs

To check for each concept in a list of concept_ids and source codes, the "isStandard.R" function is used. This function takes as input a list of concept_ids and source codes and returns a table of non-standard concepts. Saved results include all concepts regardless of standardness. See inst/extdata/phems-variable-list for examples of CSVs tables in the expected format.

```r
# Connect to DB
con <- connect(connectionDetails)
```

```
## Connecting using PostgreSQL driver
```

```r
# Return table of non-standard concepts
source('./R/isStandard.R')
nonStandard <- isStandard(
  db_connection = con,
  data_concepts_path = config_oth$concepts_path,
  vocab_schema = connectionConfig$vocabulary_schema,
  # (optional) Save the results (with standard and non-standard concepts)
  save_path = config_oth$save_path_isStandard
)
```

```
## saving file: uc1.csv
## saving file: uc2.csv
## No matches found for concept set.
##
## Finished checking for non-standard concepts.
## 17 non-standard concepts found across tables.
```

```r
# Disconnect
disconnect(con)

# Print all non-standard concepts
nonStandard
```

```
## # A tibble: 17 x 5
##    concept_id concept_name             source_code source_table standard_concept
##    <chr>      <chr>                    <chr>       <chr>        <chr>
##  1 0          No matching concept      Extracardi~ uc1.csv      Non-standard
##  2 0          No matching concept      Aristotle'~ uc1.csv      Non-standard
##  3 0          No matching concept      ventilation uc1.csv      Non-standard
##  4 0          No matching concept      Type of va~ uc1.csv      Non-standard
##  5 0          No matching concept      VIS score ~ uc1.csv      Non-standard
##  6 0          No matching concept      Total vent~ uc1.csv      Non-standard
##  7 0          No matching concept      Type of va~ uc1.csv      Non-standard
##  8 0          No matching concept      VIS score ~ uc1.csv      Non-standard
##  9 0          No matching concept      ventilation uc1.csv      Non-standard
## 10 607590     Body height              Height      uc1.csv      Non-standard
## 11 21602722   CORTICOSTEROIDS FOR SYS~ Corticoids  uc2.csv      C
## 12 21603754   Monoclonal antibodies    Monoclonal~ uc2.csv      C
## 13 21603754   Monoclonal antibodies    Monoclonal~ uc2.csv      C
## 14 0          No matching concept      Non-surgic~ uc2.csv      Non-standard
## 15 0          No matching concept      Oxygenatio~ uc2.csv      Non-standard
## 16 0          No matching concept      Urine outp~ uc2.csv      Non-standard
## 17 0          No matching concept      Leukocytes  uc2.csv      Non-standard
```

Standard and non-standard concepts given a concept set

Similarly to isStandard(), isStandardCS() can be used to check the standardness of concepts, but rather given a concept set than a list. The function returns a table of non-standard concepts. Saved results include all concepts regardless of standardness.

```
# connect to DB
con <- connect(connectionDetails)
```

```
## Connecting using PostgreSQL driver
```

```
# run for uc2 conceptSet
uc2 <- conceptSets$conceptSets$uc2

# check standardness across concept set
source('./R/isStandardCS.R')
nonStandardCS <- isStandardCS(
  conceptSet = conceptSets$conceptSets$uc2,
  # (optional) Save the results (with standard and non-standard concepts)
  save_path = config_oth$save_path_isStandard
)
```

```
## saving file: CONCEPTSET_uc2
## Found 2 non-standard concepts in concept set: uc2
```

```
# Disconnect
disconnect(con)

# print results (non-standard)
nonStandardCS
```

```
## # A tibble: 2 x 4
##   concept_name                     concept_id concept_set standard_concept
##   <chr>                                 <int> <chr>       <chr>
## 1 Monoclonal antibodies              21603754 uc2         C
## 2 CORTICOSTEROIDS FOR SYSTEMIC USE   21602722 uc2         C
```

Initial event cohort

NOTE: the cohort defined in this chunk serves solely as an example. Here we define a cohort for UC1.

```r
## Cohort definition
# Create cohort definition
ch <- cohort(
  entry = entry(
    # enter patients with conditions that would be in UC1
    conditionOccurrence(conceptSets$conceptSets$uc1),
    observationWindow = continuousObservation(0, 0),
    primaryCriteriaLimit = "All"
  ),
  attrition = attrition(
    # keep patients with all criteria matched:
    withAll(
      # criteria 1 is defined by this withAny clause:
      # keep patients with any of the following:
      withAny(
        # include patients with at least 1 cardiac complication...
        atLeast(
          x = 1,
          # with cardiac complications represented by condition concepts and...
          query = conditionOccurrence(conceptSets$conceptSets$cardiacComplications)
        ),
        atLeast(
          x = 1,
          # with cardiac complications represented by procedure concepts.
          query = procedure(conceptSets$conceptSets$cardiacComplications)
        )
      ),
      # criteria 2 is defined by the second withAny clause:
      # include patients who have had at least 1 laboratory measurement
      withAny(
        atLeast(
          x = 1,
          query = measurement(conceptSets$conceptSets$labTests)
        )
      )
    )
  ),
  # end of cohort at end of observation period
  exit = exit(
    endStrategy = observationExit()
  )
)
```

Write json expressions and sql queries

json expressions can be used to save the cohort definitions and to generate SQL queries

```
## Cohort json and sql
# Generate json for cohort
chJson <- ch %>%
  Capr::toCirce() %>%
  jsonlite::toJSON(pretty = TRUE, auto_unbox = TRUE) %>%
  as.character()

# Generate cohort sql query
sql <- CirceR::buildCohortQuery(
  expression = CirceR::cohortExpressionFromJson(chJson),
  options = CirceR::createGenerateOptions(generateStats = FALSE)
)
```

Save cohort and concept set jsons

Save the cohort and concept set jsons; these can be imported into ATLAS

```
write(chJson, paste0(config_oth$save_path_json, "/cohort.json"))
for (cs in names(conceptSets$conceptSets)) {
  writeConceptSet(
    x = conceptSets$conceptSets[[cs]],
    path = paste(config_oth$save_path_json, "/", cs, "_cs.json", sep="")
  )
}
```

Query the cohorts using the SQL query

```
# Establish connection
con <- connect(connectionDetails)
```

```
## Connecting using PostgreSQL driver
```

```
# Cohorts to create
cohortsToCreate <- tibble::tibble(
  cohortId = 9876,
  cohortName = "cohort",
  sql = sql
)

# Cohort tables
cohortTableNames <- CohortGenerator::getCohortTableNames(cohortTable = "cohort")
CohortGenerator::createCohortTables(
  connectionDetails = connectionDetails,
  cohortDatabaseSchema = "cohort",
  cohortTableNames = cohortTableNames
)
```

```
## Connecting using PostgreSQL driver

## Creating cohort tables
## - Created table cohort.cohort
## - Created table cohort.cohort
## - Created table cohort.cohort_inclusion
## - Created table cohort.cohort_inclusion_result
## - Created table cohort.cohort_inclusion_stats
## - Created table cohort.cohort_summary_stats
## - Created table cohort.cohort_censor_stats
## Creating cohort tables took 0.43secs
```

```
# Generate the cohorts
cohortsGenerated <- CohortGenerator::generateCohortSet(
  connectionDetails = connectionDetails,
  cdmDatabaseSchema = "cdm",
  cohortDatabaseSchema = "cohort",
  cohortTableNames = cohortTableNames,
  cohortDefinitionSet = cohortsToCreate
)
```

```
## Connecting using PostgreSQL driver

## Initiating cluster consisting only of main thread
## 1/1- Generating cohort: cohort (id = 9876)
##   |                                                                              |

## Executing SQL took 48.8 secs

## Generating cohort set took 49.08 secs
```

```
# Get cohort counts
cohortCounts <- CohortGenerator::getCohortCounts(
  connectionDetails = connectionDetails,
  cohortDatabaseSchema = "cohort",
  cohortTable = cohortTableNames$cohortTable
)
```

```
## Connecting using PostgreSQL driver

## Counting cohorts took 0.336 secs
```

```
# Disconnect
disconnect(con)


cohortCounts
```

```
## [1] cohortId       cohortEntries  cohortSubjects
## <0 rows> (or 0-length row.names)
```

Number of people in db

```
# Establish connection
con <- connect(connectionDetails)
```

```
## Connecting using PostgreSQL driver
```

```
# Count unique person_id in the person table
query_person <-
  paste0("SELECT COUNT(DISTINCT person_id) AS num_persons FROM ", connectionConfig$cdm_schema, ".person
result_person <- dbGetQuery(con, query_person)$num_persons

# Count unique subject_id in the cardiac_arrest table
query_cohort <-
  paste0("SELECT COUNT(DISTINCT subject_id) AS num_persons FROM ", connectionConfig$cohort_schema, ".co

result_cohort <- dbGetQuery(con, query_cohort)$num_persons

# Print results
cat("Number of persons in dataset: ", result_person, "\n")
```

```
## Number of persons in dataset:  1002
```

```
cat("Number of persons in cohort: ", result_cohort, "\n")
```

```
## Number of persons in cohort:  0
```

```
# Disconnect
disconnect(con)
```