Capr for Templating Cohort Definitions

Contents

0.1 Building a Template

Capr is at its most valuable when it is used as a tool for templating cohort definitions. In a study we may need multiple cohort definitions that use the same logic but vary by concept sets. For example, in a study we need to create cohort covariates for various conditions. The cohort logic is simple, as condition occurrence with at least 365 days of prior observation. The concept sets used could be various cardiovascular events like stroke, hypertension, acute myocardial infarction, heart failure, and atrial fibrillation.

To build all these cohorts using a template, we can easily define the logic in Capr. Capr templates are functions were the input is a concept set that differs in the logic.

```
cvEvents <- function(conceptSet) {
  cd <- cohort(
    entry = entry(
      conditionOccurrence(conceptSet),
      observationWindow = continuousObservation(365, 0)
    ),
    exit = exit(
      endStrategy = observationExit()
    )
  )
  return(cd)
}
```

The cohort template above describes the logic we desire. An entry event based on a condition occurrence with 365 days of prior observation and the cohort exit is based on the last available date of observation.

Once we have a Capr template, our next step is to add concept sets into the template functions. Using Athena, I look up the OMOP concept ids for the concepts of interest. With these concept ids, I can create a concept set for each cardiovascular event

```
afib <- cs(descendants(313217), name = "Atrial Fibrillation")
stroke <- cs(descendants(4310996), name = "Ischemic Stroke")
hyp <- cs(descendants(320128), name = "Hypertension")
mi <- cs(descendants(4329847), name = "Myocardial Infarction")
hf <- cs(descendants(316139), name = "Heart Failure")
```

Once I have my concept sets, I could build each cohort one at a time. What is preferred is to use the R functional factories in order to not write code for each concept set. It is recommended to use the function factories provided in the `purrr` package. Below is an example of how we can use purrr to build multiple cohorts from our Capr template. The `cvCohorts` object returns a list of four Capr Cohort class objects.

```
cvSet <- list(afib, stroke, hyp, mi, hf)
cvCohorts <- purrr::map(cvSet, ~cvEvents(.x))
```

0.2 Improving the Template

In the previous example, our `cvEvents` function returned a Capr Cohort object. What if we wanted the output to be json and even better save to a folder? We can improve the Capr template function to create the json object; an example is provided below.

```
cvEvents2 <- function(conceptSet) {

  #Capr template logic
  cd <- cohort(
    entry = entry(
      conditionOccurrence(conceptSet),
      observationWindow = continuousObservation(365, 0)
    ),
    exit = exit(
      endStrategy = observationExit()
    )
  )

  #coerce cohort to json
  cohortJson <- cd %>%
    toCirce() %>%
    jsonlite::toJSON(pretty = TRUE, auto_unbox = TRUE) %>%
    as.character()

  return(cohortJson)
}
```

0.3 Building templates from concept sets

Sometimes a single concept is insufficient for defining a clinical idea and we need multiple concepts in the set. Also we may want to use a concept set developed previously. The Capr function `readConceptSet` allows one to import a concept set from a .csv or .json file that we can use towards the cohort template. We update our Capr template function to handle the import of a concept set, as shown below.

```
cvEvents3 <- function(file) {

  # get file name
  name <- tools::file_path_sans_ext(basename(file))

  #retreive concept set
  conceptSet <- Capr::readConceptSet(path = file, name = name)

  #Capr template logic
  cd <- cohort(
    entry = entry(
      conditionOccurrence(conceptSet),
      observationWindow = continuousObservation(365, 0)
    ),
    exit = exit(
      endStrategy = observationExit()
```

```
    )
  )

  #coerce cohort to json
  cohortJson <- cd %>%
    toCirce() %>%
    jsonlite::toJSON(pretty = TRUE, auto_unbox = TRUE) %>%
    as.character()

  return(cohortJson)
}
```

Let's use the example of acute myocardial infarction. For the cohort, we want to use a concept set that includes MI but excludes old MI. Luckily, we have a concept set from ATLAS that we used in a previous study that has this already detailed. We can import the csv file for this concept set and apply it to the Capr template.

```
miPath <- fs::path_package("Capr", "extdata/acuteMI.csv")
miCohort <- cvEvents3(miPath)
cat(miCohort)
```

0.4 Saving Capr cohorts

Once you have built the cohorts you need for your study using Capr, you want to save them to a directory to use them in the study. Taking the `miCohort` as an example, we can save the json output to file using `readr::write_file` or the base R equivalent.

```
outputPath <- fs::path(here::here("cohorts"), "miCohort", ext = "json")
readr::write_file(miCohort, file = outputPath)
```

To save multiple cohorts from a list we can functionalize the `readr::write_file` using `purrr::walk`.

0.5 Final thoughts

Cohort templating is a very powerful application of Capr. Instead of defining the cohort logic for multiple inputs, we can create a template to iterate across inputs. Your template can be as simple or as complex as you wish. By following the Capr syntax, a user can build cohorts how they please and iterate! The templates become more manageable when using function factories in R offered by the `purrr` package.