Using Template Cohorts

James P. Gilbert

2025-11-11

Contents

1 Introduction

This guide intends to demonstrate the usage of template cohorts within the Cohort Generator package. This can provide a convenient approach to computing large sets of features. While this is possible through the use of custom scripts, doing so will often require one-off approaches to integrating references within studies or other OHDSI packages, greatly limiting their reproducibility.

The principle behind this implementation is that, for all intents and purposes, cohorts created via "bulk" operations should be treated no differently to cohorts created through Circe definitions.

1.1 Limitations of this approach

For the design of reliable, reusable Phenotype Algorithms, we strongly advise the usage of Circe based approaches. While there is a trade-off that such an approach may be less efficient that pure SQL, this will greatly limit the reproducibility and replicability of studies using these cohorts.

2 Basic SQL templates

The most straightforward cohort template is an SQL only definition for an individual cohort. A good use case for this example is a cohort for which a Circe definition is inefficient and an equivalent cohort can be made with SQL alone or, in this example, where the cohort is speculative based on the standard vocabulary. Here we create a cohort that searches for all drug exposusres based on string patterns in the concept vocabulary table.

The first step is to create a definition as follows

```r
library(CohortGenerator)

cohortDefinitionSet <- createEmptyCohortDefinitionSet()

sql <- "INSERT INTO @cohort_database_schema.@cohort_table
              (cohort_definition_id, subject_id, cohort_start_date, cohort_end_date)
         SELECT 1 as cohort_definition_id,
                person_id as subject_id,
                drug_era_start_date as cohort_start_date,
                drug_era_end_date as cohort_end_date
         FROM @cdm_database_schema.drug_era de
         INNER JOIN @cdm_database_schema.concept c on de.drug_concept_id = c.concept_id
         -- Find any matches of drugs named 'asprin' in the drug concept table
         WHERE lower(c.concept_name) like '%asprin%'; "

cohortDefinitionSet <- cohortDefinitionSet |>
  addSqlCohortDefinition(
    sql = sql,
    cohortId = 1,
    cohortName = "my asprin cohort"
  )

connection <- DatabaseConnector::connect(Eunomia::getEunomiaConnectionDetails())
```

```
## Connecting using SQLite driver
```

```r
createCohortTables(connection = connection, cohortDatabaseSchema = "main")
```

```
## Creating cohort tables
## - Created table main.cohort
## - Created table main.cohort
## - Created table main.cohort_inclusion
## - Created table main.cohort_inclusion_result
## - Created table main.cohort_inclusion_stats
## - Created table main.cohort_summary_stats
## - Created table main.cohort_censor_stats
## - Created table main.cohort_checksum
## Creating cohort tables took 0.1secs
```

```r
status <- generateCohortSet(
  connection = connection,
  cdmDatabaseSchema = "main",
  cohortTableNames = getCohortTableNames(),
  cohortDefinitionSet = cohortDefinitionSet,
  incremental = TRUE
)
```

```
## Generating Template Cohort: my asprin cohort

## Warning: Parameter 'vocabulary_database_schema' not found in SQL

## Template Cohort complete: my asprin cohort
```

Note - in this definition we do not ever reference the bound variables `@cohort_table`, `@cohort_database_schema`, `@cdm_database_schema` or `@vocabulary_database_schema`. This is for two reasons, firstly these are already known at the time of cohort generation, secondly they will make the cohort definition not generalizable to other data sources (i.e. the phenotype algorithm will not be transportable to other data sets). This may also

leak information about your CDM and should be checked before inclusion within network studies.

2.0.1 Validating custom sql cohorts

To be valid, a cohort definition must:

- Include a cohort id, subject id, start date and end date
- The start date must always be on or before the end date
- There must be no overlapping or duplicate eras for the same subject within a cohort
- In general, cohorts should be only inside observation periods within the CDM

With Circe based cohorts, it is guaranteed that definitions meet this criteria. However, this is not necessarily true when generating custom SQL definitions. Consequently, we use the `getCohortValidationCounts` function within the `CohortGenerator` package.

```
getCohortValidationCounts(
  connection = connection,
  cdmDatabaseSchema = "main",
  cohortDatabaseSchema = "main"
)
```

```
## Computing cohort validation checks
## Computed validation checks for 0 cohorts
## Generating validation check set took 0.03 secs
```

```
## [1] cohortDefinitionId           overlappingErasCount         invalidDateCount             duplicate
## [5] outsideObservationStartCount outsideObservationEndCount   valid
## <0 rows> (or 0-length row.names)
```

3 Built in large scale definitions

There are currently 3 large scale cohort generation methods included within the package that use vocabulary tables to bulk generate thousands of cohorts in a time that would be infieasible if using Circe standard cohorts from ATLAS or Capr. These can be broadly defined as:

1. `createRxNormCohortTemplateDefinition` This is definition of all RxNorm standard ingredient concepts that uses the concept ancestor table to merge all definitions into a single cohort.
2. `createAtcCohortTemplateDefinition` This is definition of all ATC standard ingredients using the concept hierarchy. This can either be generated from the first exposure or from merging all eras between individual ingredients
3. `createSnomedCohortTemplateDefinition` This a definition of all SNOMED (OHDSI standard) condition occurences (with some exclusions based on strings to exclude)

3.1 Drug ingredient cohorts

```
# Library imports
library(CohortGenerator)
library(DatabaseConnector)

# Create RxNorm ingredient cohort template
rxNormDefinition <- createRxNormCohortTemplateDefinition(
  connection = connection, # Replace with your DatabaseConnector connection
  cdmDatabaseSchema = "main",
  priorObservationPeriod = 365
)
```

```r
cohortDefinitionSet <- cohortDefinitionSet |>
  addCohortTemplateDefintion(cohortTemplateDefintion = rxNormDefinition)


# View details of generated template references
rxNormReferences <- rxNormDefinition$getTemplateReferences()
head(rxNormReferences)
```

```
##     cohortId               cohortName json                                                                sql is
## 1 1557272000     RxNorm - Alendronate   {} SELECT '1557272000 - All RxNorm ingredient exposures';
## 2  708298000       RxNorm - Midazolam   {}  SELECT '708298000 - All RxNorm ingredient exposures';
## 3  701322000       RxNorm - Memantine   {}  SELECT '701322000 - All RxNorm ingredient exposures';
## 4  723013000        RxNorm - Diazepam   {}  SELECT '723013000 - All RxNorm ingredient exposures';
## 5 1129625000 RxNorm - Diphenhydramine   {} SELECT '1129625000 - All RxNorm ingredient exposures';
## 6 1149196000      RxNorm - Cetirizine   {} SELECT '1149196000 - All RxNorm ingredient exposures';
```

3.2 ATC Base cohorts

```r
# Create ATC-based cohort template
atcDefinition <- createAtcCohortTemplateDefinition(
  connection = connection, # Replace with your DatabaseConnector connection
  cdmDatabaseSchema = "main",
  mergeIngredientEras = TRUE,
  priorObservationPeriod = 365
)

cohortDefinitionSet <- cohortDefinitionSet |>
  addCohortTemplateDefintion(cohortTemplateDefintion = atcDefinition)

# View ATC template references
atcReferences <- atcDefinition$getTemplateReferences()
head(atcReferences)
```

3.3 SNOMED condition cohorts

```r
# Create SNOMED cohort template
snomedDefinition <- createSnomedCohortTemplateDefinition(
  connection = connection, # Replace with your DatabaseConnector connection
  cdmDatabaseSchema = "main",
  priorObservationPeriod = 180 # Require 180 days prior observation
)

cohortDefinitionSet <- cohortDefinitionSet |>
  addCohortTemplateDefintion(cohortTemplateDefintion = snomedDefinition)

# View the condition template references
snomedReferences <- snomedDefinition$getTemplateReferences()
head(snomedReferences)
```

```
##     cohortId                                cohortName json
## 1 4116491000 Escherichia coli urinary tract infection   {} SELECT '4116491000 - All SNOMED Conditions
## 2 4113008000                        Laceration of hand   {} SELECT '4113008000 - All SNOMED Conditions
## 3 4156265000                         Facial laceration   {} SELECT '4156265000 - All SNOMED Conditions
```

```
## 4 4155034000                    Laceration of forearm   {} SELECT '4155034000 - All SNOMED Conditions
## 5 4109685000                       Laceration of foot   {} SELECT '4109685000 - All SNOMED Conditions
## 6 4094814000                             Bullet wound   {} SELECT '4094814000 - All SNOMED Conditions
```

4 Creating custom cohort templates

Creating custom templates involves a number of developer decisions that must be decided: 1. How will references be defined?

This must be a `data.frame` that contains `cohortId` and `cohortName` fields. Additionally, this may contain json (for example, if you wish to relate all templates to a Circe definition that includes concept sets). In all the 3 methods above, templates are generated entirely from the vocabulary within an OMOP CDM but this detail is left to be implementation specific.

2. Define an parameterize SQL.

The SQL for generating template cohorts has a few restrictive properties: * It must result in inserts into the cohort table with column names `cohort_definition_id`, `cohort_start_date`, `cohort_end_date`, `subject_id` * It should use the SqlRender standard bindings `@cohort_table`, `@cdm_database_schema`, `@cohort_database_schema` and `@vocabulary_database_schema` (in general vocabulary tables should be stored within the cdm).

4.1 Generating the cohorts

Template cohorts generate in the standard model within cohort generator.

```
status <- generateCohortSet(
  connection = connection,
  cdmDatabaseSchema = "main",
  cohortTableNames = getCohortTableNames(),
  cohortDefinitionSet = cohortDefinitionSet,
  incremental = TRUE
)
```

```
## Skipping Template Cohort: my asprin cohort
## Generating Template Cohort: All RxNorm ingredient exposures
```

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

```
## Template Cohort complete: All RxNorm ingredient exposures
## Generating Template Cohort: All SNOMED Conditions
## Template Cohort complete: All SNOMED Conditions
```

4.1.1 On execution order

One note on execution is that the order of execution is:

1. Circe cohorts
2. Non-standard sql/Template sql cohorts
3. Subset cohort definitions

For this reason, it is possible to both create subset based cohorts form templates. However, it is not currently possible to reference subsetted cohorts from template definitions under this standard execution framework. When referencing Circe cohorts in template definitions, special care should be taken around incremental execution.

5 Conclusion

The CohortGenerator package offers a robust, flexible approach for creating large-scale or custom cohorts, making it easier to prototype and analyze in the OHDSI ecosystem. By leveraging built-in templates or user-defined SQL, you can create cohorts that combine efficiency, reproducibility, and scalability.

Key Takeaways: * Large-scale cohort templates (RxNorm, ATC, SNOMED) allow for rapid, efficient generation of thousands of cohorts. * Custom SQL-based templates enable precise control and address unique research requirements. * The approach integrates smoothly with OHDSI tools like ATLAS, CAPR, and DatabaseConnector to ensure compatibility across the OHDSI ecosystem.