Generating Cohorts

Anthony G. Sena

2021-12-06

Contents

1 Guide for generating cohorts using CohortGenerator

This guide will provide examples of using the CohortGenerator package to generate cohorts in R. In this vignette, we will walk through the process of downloading cohorts from ATLAS and look at the options available for generating the cohorts.

1.1 Basic Example

1.1.1 Downloading cohorts from ATLAS

We can create cohorts using ATLAS and download them for generation in R. To do this, we will use the `ROhdsiWebApi` package to download some cohort definitions to R:

```
# Connect to WebAPI and retrieve the full list of cohorts
cohortDefinitions <- ROhdsiWebApi::getCohortDefinitionsMetaData(baseUrl = baseUrl)
# Filter the list to the cohorts for this example
cohortsForGeneration <- cohortDefinitions[grepl(pattern = "^\\[CohortGenerator\\]", cohortDefinitions$na
# Get the SQL/JSON for the cohorts
cohortDefinitionSet <- ROhdsiWebApi::exportCohortDefinitionSet(baseUrl = baseUrl,
                                                               cohortIds = cohortsForGeneration$id,
                                                               generateStats = FALSE)
```

The code above connects to ATLAS's WebAPI, retrieves all of the `cohortDefinitions` and then filters them based on the name to produce the `cohortsForGeneration` data frame. `cohortsForGeneration` contains the list of cohort IDs to use for calling the function `ROhdsiWebApi::exportCohortDefinitionSet` to download the set of cohort definitions into `cohortDefinitionSet`. The cohort definition set data frame has the following columns:

```
names(cohortDefinitionSet)
```

```
#> [1] "atlasId"          "cohortId"         "cohortName"       "sql"              "json"             "
```

Here is how these columns are used:

- **atlasId**: The ATLAS ID for the cohort. This provides linkage back to the ATLAS cohort definition
- **cohortId**: The cohortId will be the same as the atlasId upon export from ATLAS. We provide this column in case you'd like to alter the numbering scheme for your cohort definition set.
- **cohortName**: The name of the cohort in ATLAS.
- **sql**: The SQL used to construct the cohort.

- **json**: The Circe compliant JSON representation of the cohort definition
- **logicDescription**: The description of the cohort from ATLAS.

1.1.2 Generating Cohorts

Now that we have obtained the `cohortDefinitionSet` from ROhdsiWebApi, we're ready to generate our cohorts against our OMOP CDM. In this example, we will use the Eunomia data set as our CDM.

```
# Get the Eunomia connection details
connectionDetails <- Eunomia::getEunomiaConnectionDetails()

# First get the cohort table names to use for this generation task
cohortTableNames <- CohortGenerator::getCohortTableNames(cohortTable = "cg_example")

# Next create the tables on the database
CohortGenerator::createCohortTables(connectionDetails = connectionDetails,
                                    cohortTableNames = cohortTableNames,
                                    cohortDatabaseSchema = "main",
                                    incremental = FALSE)
```

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

```
#> Creating cohort tables
#> - Created table main.cg_example
#> - Created table main.cg_example_inclusion
#> - Created table main.cg_example_inclusion_result
#> - Created table main.cg_example_inclusion_stats
#> - Created table main.cg_example_summary_stats
#> - Created table main.cg_example_censor_stats
#> Creating cohort tables took 0.46secs
```

```
# Generate the cohort set
CohortGenerator::generateCohortSet(connectionDetails= connectionDetails,
                                   cdmDatabaseSchema = "main",
                                   cohortDatabaseSchema = "main",
                                   cohortTableNames = cohortTableNames,
                                   cohortDefinitionSet = cohortDefinitionSet,
                                   incremental = FALSE,
                                   incrementalFolder = file.path(folderName, "RecordKeeping"))
```

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

```
#> 1/3- Generating cohort: [CohortGenerator] celecoxib
#>   |
```

```
#> 1  1778211                  [CohortGenerator] celecoxib         COMPLETE 2021-12-06 14:10:25 2021-12-
#> 2  1778212          [CohortGenerator] celecoxib age 40         COMPLETE 2021-12-06 14:10:25 2021-12-
#> 3  1778213 [CohortGenerator] celecoxib age 40 and male         COMPLETE 2021-12-06 14:10:25 2021-12-
```

The code above starts by obtaining the `connectionDetails` from Eunomia. This is where you'll likley want to substitute your own connection information. Next, we call `getCohortTableNames` to obtain a list of `cohortTableNames` that we'll use for the generation process. We then call `createCohortTables` to create the cohort tables on the database server in the `cohortDatabaseSchema`. Once these tables are created, we use the function `generateCohortSet` to generate the `cohortDefintionSet`. When calling `generateCohortSet`, we must specify the schema that holds our CDM (`cdmDatabaseSchema`), the location of the `cohortDatabaseSchema` and `cohortTableNames` where the cohort(s) will be generated. We will cover the other parameters available in the Advanced Options section.

If we'd like to see the results of the generation process, we can use the `getCohortCounts` method to query the cohort table for a summary of the persons and events for each cohort:

```
cohortCounts <- CohortGenerator::getCohortCounts(connectionDetails = connectionDetails,
                                                 cohortDatabaseSchema = "main",
                                                 cohortTable = cohortTableNames$cohortTable)
```

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

```
#> Counting cohorts took 0.068 secs
```

```
cohortCounts
```

```
#>   cohortId cohortEntries cohortSubjects
#> 1  1778211          1800           1800
#> 2  1778212           569            569
#> 3  1778213           266            266
```

1.2 Advanced Options

1.2.1 Cohort Statistics (Inclusion Rule Statistics)

TODO

1.2.2 Incremental Options

TODO