Running Cohort Diagnostics

Gowtham Rao and James P. Gilbert

2022-06-29

Contents

1 Introduction

This vignette discusses the process of generating a results set with `CohortDiagnostics` starting with cohort generation. Please see the HADES library for more information on the background for this.

1.1 pre-requisites

Ensure that `CohortDiagnostics` is installed on your system and updated to the latest version. For this example we will also be using the `Eunomia` test package. Optionally, you may install the `ROhdsiWebApi` package to download cohort definitions from an ATLAS instance:

```
remotes::install_github('OHDSI/Eunomia')
remotes::install_github('OHDSI/ROhdsiWebApi')
```

2 Configuring the connection to the server

We need to tell R how to connect to the server where the data are. `CohortDiagnostics` uses the `DatabaseConnector` package, which provides the `createConnectionDetails` function. Type `?createConnectionDetails` for the specific settings required for the various database management systems (DBMS). For example, one might connect to a PostgreSQL database using this code:

```
library(CohortDiagnostics)

connectionDetails <- createConnectionDetails(dbms = "postgresql",
                                             server = "localhost/ohdsi",
                                             user = "joe",
                                             password = "supersecret")
```

For the purposes of this example, we will use the Eunomia test CDM package that is in an Sqlite database stored locally.

```
connectionDetails <- Eunomia::getEunomiaConnectionDetails()

cdmDatabaseSchema <- "main"
tempEmulationSchema <- NULL
cohortDatabaseSchema <- "main"
cohortTable <- "cohort"
```

The last four lines define the `cdmDatabaseSchema`, `tempEmulationSchema`, `cohortDatabaseSchema`, and `cohortTable` variables. We'll use the `cdmDatabaseSchema` later to tell R where the data in CDM format live. The `tempEmulationSchema` is needed only for Oracle users, since Oracle does not support temporary tables. The `cohortDatabaseSchema`, and `cohortTable` specify where we want to instantiate our cohorts. Note that for Microsoft SQL Server, database schemas need to specify both the database and the schema, so for example `cdmDatabaseSchema <- "my_cdm_data.dbo"`.

3 Running cohort diagnostics

3.1 Loading cohort references from a package

The prefered usage of cohort diagnostics is through the use of a study package. This is a dedicated R package that can be installed on a system and run. The primary reason for this is due to reproducibility, cohort definitions and resources frequently change. However, a study package can be seen as a snapshot, frozen at the time of creation and incrementally updated.

For example, the cohort diagnostics package includes an example set of cohort sql and json to run on the `Eunomia` test data in the OMOP Common Data Model format.

```
library(CohortDiagnostics)
cohortDefinitionSet <- CohortGenerator::getCohortDefinitionSet(settingsFileName = "Cohorts.csv",
                                                               jsonFolder = "cohorts",
                                                               sqlFolder = "sql/sql_server",
                                                               packageName = "CohortDiagnostics")
```

Looking at this data.frame of Cohorts you will see the sql and json for these cohorts:

```
View(cohortDefinitionSet)
```

3.2 Loading cohort references from WebApi

It is often desirable to perform cohort diagnostics on definitions stored in an ATLAS instance. Though this is not the preferred way of running studies (and this is certainly not the preferred method for an OHDSI network study involving multiple sites) it is possible to load references into a data frame used by cohort diagnostics.

The following code demonstrates how to create a set of cohort references from ATLAS that can be used by cohort diagnostics:

```
# Set up url
baseUrl <- "https://atlas.hosting.com/WebAPI"
# list of cohort ids
cohortIds <- c(18345,18346)

cohortDefinitionSet <- ROhdsiWebApi::exportCohortDefinitionSet(baseUrl = baseUrl,
                                                               cohortIds = cohortIds,
                                                               generateStats = TRUE)
```

Consult the ROhdsiWebApi documentation for details on authentication to your atlas instance. Please note that in order to generate inclusion rules statistics (a useful diagnostic tool) the parameter `generateStats` should be set to `TRUE`.

3.3 Generating cohorts

Cohorts must be generated before cohort diagnostics can be run.

3.3.1 Using CohortGenerator to instantiate cohorts

For example,

```
cohortTableNames <- CohortGenerator::getCohortTableNames(cohortTable = cohortTable)

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

# Generate the cohort set
CohortGenerator::generateCohortSet(connectionDetails= connectionDetails,
                                   cdmDatabaseSchema = cdmDatabaseSchema,
                                   cohortDatabaseSchema = cohortDatabaseSchema,
                                   cohortTableNames = cohortTableNames,
                                   cohortDefinitionSet = cohortDefinitionSet,
                                   incremental = FALSE)
```

Note, that the above code will delete an existing table. However, incremental mode can be used when setting the parameter `incremental = TRUE`.

The resulting cohort table should include the columns:

`cohort_definition_id`, `subject_id`, `cohort_start_date`, `cohort_end_date`

3.4 Executing cohort diagnostics

Once cohort definitions are loaded and cohort tables have been populated cohort diagnostics is ready to be executed.

First we set an export folder, this is where the results will be stored.

```
exportFolder <- "export"
```

Then we execute the function (using the default settings) as follows:

```
executeDiagnostics(cohortDefinitionSet,
                   connectionDetails = connectionDetails,
                   cohortTable = cohortTable,
                   cohortDatabaseSchema = cohortDatabaseSchema,
                   cdmDatabaseSchema = cdmDatabaseSchema,
                   exportFolder = exportFolder,
                   databaseId = "MyCdm",
                   minCellCount = 5)
```

3.5 Cohort Statistics Table Clean up

The above cohort generation process will create a number of residual tables. As the process is complete, these are no longer required and can be removed.

```
CohortGenerator::dropCohortStatsTables(connectionDetails = connectionDetails,
                                       cohortDatabaseSchema = cohortDatabaseSchema,
                                       cohortTableNames = cohortTableNames)
```

4 Cohort Diagnostics Output

Once the diagnostics have completed, a zip file will have been created in the specified export folder. This zip file can be shared between sites, as it does not contain patient-identifiable information. When unzipped, the zip file will contain several .csv files that maybe easily audited. Note that cell counts smaller than 5 have been removed, as specified using the `minCellCount` argument, to ensure non-identifiability.

4.1 Creating an sqlite db file

Assuming you completed the steps described above for one or more databases, you should now have a set of zip files, one per database. Make sure to place all zip files in a single folder.

Optionally, we can pre-merge the zip files into an sqlite database so we can view results in the Shiny app:

```
createMergedResultsFile(exportFolder)
```

This file can be used in the shiny app to explore results. See the vignette "Viewing results using Diagnostics Explorer" for more details.