

# Running cohort diagnostics using WebAPI

Gowtham Rao

2021-11-13

## Contents

|          |                                                     |          |
|----------|-----------------------------------------------------|----------|
| <b>1</b> | <b>Introduction</b>                                 | <b>1</b> |
| <b>2</b> | <b>Running the cohort diagnostics</b>               | <b>1</b> |
| 2.1      | Defining the set of cohorts to diagnose . . . . .   | 1        |
| 2.2      | Instantiating the cohorts . . . . .                 | 2        |
| 2.3      | Generating the diagnostics in WebApi mode . . . . . | 3        |
| 2.4      | Incremental mode . . . . .                          | 3        |

## 1 Introduction

There are currently two approaches to run Cohort Diagnostics. - Embed in an OHDSI study package, where all the cohort definitions are stored as part of that study package, or - WebApi mode - where cohort diagnostics dynamically pulls the cohort definition from a webapi instance. WebAPI is the backend of the OHDSI ATLAS application, allowing programmatic access to the cohort definitions created in ATLAS.

This vignette describes the latter approach (webapi): how to run CohortDiagnostics using the WebAPI.

## 2 Running the cohort diagnostics

### 2.1 Defining the set of cohorts to diagnose

The first step is to define the set of cohorts we wish to create diagnostics for. We do this by creating a data frame with four columns:

- **atlasId**: The cohort ID in ATLAS.
- **atlasName**: The full name of the cohort. This will be shown in the Shiny app.
- **cohortId**: The cohort ID to use in the package. Usually the same as the cohort ID in ATLAS.
- **name**: A short name for the cohort, to use to create file names. do not use special characters.

One way to create such a data frame is to create a CSV file, and load it into R. Here is an example table we assume is stored in `cohortsToDiagnose.csv`:

| atlasId | atlasName                                                                       | cohortId | name           |
|---------|---------------------------------------------------------------------------------|----------|----------------|
| 1770710 | New users of ACE inhibitors as first-line monotherapy for hypertension          | 1770710  | ace_inhibitors |
| 1770711 | New users of Thiazide-like diuretics as first-line monotherapy for hypertension | 1770711  | thz            |
| 1770712 | Angioedema outcome                                                              | 1770712  | angioedema     |

| atlasId | atlasName                              | cohortId | name |
|---------|----------------------------------------|----------|------|
| 1770713 | Acute myocardial infarction<br>outcome | 1770713  | ami  |

We can read the table using

```
library(CohortDiagnostics)
cohortSetReference <- read.csv("cohortsToDiagnose.csv")
```

An easy way to create this csv file is to use ROhdsiWebApi

```
library(magrittr)
# Set up
baseUrl <- Sys.getenv("BaseUrl")
# list of cohort ids
cohortIds <- c(18345,18346)

# get specifications for the cohortIds above
webApiCohorts <-
  ROhdsiWebApi::getCohortDefinitionsMetaData(baseUrl = baseUrl) %>%
  dplyr::filter(.data$id %in% cohortIds)

cohortsToCreate <- list()
for (i in (1:nrow(webApiCohorts))) {
  cohortId <- webApiCohorts$id[[i]]
  cohortDefinition <-
    ROhdsiWebApi::getCohortDefinition(cohortId = cohortId,
                                       baseUrl = baseUrl)
  cohortsToCreate[[i]] <- tidyr::tibble(
    atlasId = webApiCohorts$id[[i]],
    atlasName = stringr::str_trim(stringr::str_squish(cohortDefinition$name)),
    cohortId = webApiCohorts$id[[i]],
    name = stringr::str_trim(stringr::str_squish(cohortDefinition$name))
  )
}
cohortsToCreate <- dplyr::bind_rows(cohortsToCreate)

readr::write_excel_csv(x = cohortsToCreate, na = "",
                      file = "D:/temp/CohortsToCreate.csv",
                      append = FALSE)
```

See Vignette on ‘Running Cohort Diagnostics’ on how to connect cohort diagnostics to CDM and creating cohort table.

## 2.2 Instantiating the cohorts

To instantiate the cohorts we specified in the `cohortSetReference` we need to communicate with the WebAPI instance, as well as the database server.

To connect to the WebAPI, we need to provide the base URL. This is a URL that looks something like “http://server.org:80/WebAPI”. If you do not know the WebAPI’s base URL, contact the ATLAS administrator. Note: there is no trailing ‘/’.

To instantiate the cohorts:

```

baseUrl <- "http://server.org:80/WebAPI"
inclusionStatisticsFolder <- "c:/temp/incStats"

instantiateCohortSet(connectionDetails = connectionDetails,
                     cdmDatabaseSchema = cdmDatabaseSchema,
                     tempEmulationSchema = tempEmulationSchema,
                     cohortDatabaseSchema = cohortDatabaseSchema,
                     cohortTable = cohortTable,
                     baseUrl = baseUrl,
                     cohortSetReference = cohortSetReference,
                     generateInclusionStats = TRUE,
                     inclusionStatisticsFolder = inclusionStatisticsFolder)

```

This command will contact the WebApi to obtain the cohort definitions, instantiate them in the cohort table, and write the inclusion rule statistics to the specified folder.

## 2.3 Generating the diagnostics in WebApi mode

Next we generate the cohort diagnostics:

```

databaseId <- "MyData"
exportFolder <- "c:/temp/export"

runCohortDiagnostics(baseUrl = baseUrl,
                     cohortSetReference = cohortSetReference,
                     connectionDetails = connectionDetails,
                     cdmDatabaseSchema = cdmDatabaseSchema,
                     tempEmulationSchema = tempEmulationSchema,
                     cohortDatabaseSchema = cohortDatabaseSchema,
                     cohortTable = cohortTable,
                     inclusionStatisticsFolder = inclusionStatisticsFolder,
                     exportFolder = exportFolder,
                     databaseId = databaseId,
                     runInclusionStatistics = TRUE,
                     runIncludedSourceConcepts = TRUE,
                     runOrphanConcepts = TRUE,
                     runBreakdownIndexEvents = TRUE,
                     runIncidenceRate = TRUE,
                     runCohortOverlap = TRUE,
                     runCohortCharacterization = TRUE,
                     minCellCount = 5)

```

The databaseId is a short string that will be used to identify the data from this database in the Shiny app. Make sure to give it a name you can easily recognize.

## 2.4 Incremental mode

Although possible, we don't recommend using incremental mode in WebApi mode, as the cohort definitions may change in the target webapi instance.