Running Achilles on Your CDM

Ajit Londhe

2021-03-03

Contents

1 Introduction

In this vignette we cover how to run the Achilles package on your Common Data Model (CDM) database in order to characterize the dataset and run data quality (DQ) checks. The characterizations and DQ results can help you learn more about your dataset's features and limitations, and can then be consumed graphically using AchillesWeb or Atlas Data Sources.

It is a best practice for all OHDSI sites to run Achilles on their CDM datasets to ensure researchers can evaluate study feasibility and contextualize study results.

2 General Approach

The Achilles package consists of:

1. The **achilles** function runs a set of SQL scripts to characterize the domains and concepts of the CDM.
2. The **createIndices** function creates table indices for the achilles tables, which can help improve query performance.
3. The **getAnalysisDetails** function provides descriptions about the full set of Achilles analyses.
4. The **dropAllScratchTables** function is useful only for multi-threaded mode. It can clear any leftover staging tables.
5. The **exportToJson** function can be used to export all Achilles results to JSON files, which is necessary for using AchillesWeb.
6. The **addDataSource** function can point a data source's JSON files to the AchillesWeb application.

2.1 SQL Only Mode

In most Achilles functions, you can specify `sqlOnly = TRUE` in order to produce the SQL without executing it, which can be useful if you'd like to examine the SQL closely or debug something. The SQL files are stored in the `outputFolder`.

2.2 Logging

File and console logging is enabled across most Achilles functions. The status of each step is logged into files in the `outputFolder`. You can review the files in a common text editor.

2.3 Verbose Mode

The `verboseMode` parameter can be set to FALSE if you'd like less details about the function execution to appear in the console. Either way, all details are written to the log files. By default, this is set to TRUE.

2.4 Preparation for running Achilles

In order to run the package, you will need to determine if you'd like the Achilles tables and staging tables to be stored in schemas that are separate from your CDM's schema (recommended), or within the same schema as the CDM.

2.4.1 Multi-Threaded vs Single-Threaded

As the **achilles** and most of the **achillesHeel** functions can run independently, we have added a multi-threaded mode to allow for more than 1 SQL script to execute at a time. This is particularly useful for massively parallel processing (MPP) platforms such as Amazon Redshift and Microsoft PDW. It may not be beneficial for traditional SQL platforms, so only use the multi-threaded mode if confident it can be useful.

Further, while multiple threads can help performance in MPP platforms, there can be diminishing returns as the cluster has a finite number of concurrency slots to handle the queries. A rule of thumb: most likely you should not use more than 10.

In the multi-threaded mode, all scripts produce permanent staging tables, whereas in the single-threaded mode, the scripts produce temporary staging tables. In both, the staging tables are merged to produce the final Achilles tables.

3 Achilles Parameters (Both Modes)

The following sub-sections describe the optional parameters in **achilles** that can be configured, regardless of whether you run the function in single- or multi-threaded mode.

3.1 Staging Table Prefix

To keep the staging tables organized, the **achilles** function will use a table prefix of "tmpach" by default, but you can choose a different one using the `tempAchillesPrefix` parameter. This is useful for database platforms like Oracle, which limit the length of table names.

3.2 Source Name

The `sourceName` parameter is used to assign the name of the dataset to the Achilles results. It is used in the Dashboard page in AchillesWeb and Atlas Data Sources. If you set this to `NULL`, the **achilles** function will try to obtain the source name from the CDM_SOURCE table.

3.3 Create Table

The `createTable` parameter, when set to `TRUE`, drops any existing Achilles results tables and builds new ones. If set to `FALSE`, these tables will persist, and the **achilles** function will just insert new data to them.

3.4 Limiting the Analyses

By default, the **achilles** function runs all analyses detailed in the `getAnalysisDetails` function. However, it may be useful to focus on a subset of analyses rather than running the whole set. This can be accomplished by specifying analysis Ids in the `analysisIds` parameter.

3.5 Cost Analyses

By default, the **achilles** function does not run analyses on the COST table(s), as they can be very time-consuming, and are not critical to most OHDSI studies. However, you can choose to run these analyses by setting `runCostAnalysis` to `TRUE`. The cost analyses are conditional on the CDM version. If using CDM v5.0, then the older cost tables are queried. If using any version after 5.0, the unified cost table is queried.

3.6 Small Cell Count

To avoid patient identification, you can establish the minimum cell size that should be kept in the Achilles tables. Cells with small counts (less than or equal to the value of the `smallCellCount` parameter) are deleted. By default, this is set to 5. Set to 0 for complete summary without small cell count restrictions.

3.7 Drop Scratch Tables

See the Post-Processing section to read about how to run this step separately

This parameter is only necessary if running in multi-threaded mode

The `dropScratchTables` parameter, if set to `TRUE`, will drop all staging tables created during the execution of **achilles** in multi-threaded mode.

3.8 Create Indices

See the Post-Processing section to read about how to run this step separately

The `createIndices` parameter, if set to `TRUE`, will result in indices on the Achilles results tables to be created in order to improve query performance.

3.9 Return Value

When running **achilles**, the return value, if you assign a variable to the function call, is a list object in which metadata about the execution and all of the SQL scripts executed are attributes. You can also run the function call without assigning a variable to it, so that no values are printed or returned.

4 Running Achilles: Single-Threaded Mode

In single-threaded mode, there is no need to set a `scratchDatabaseSchema`, as temporary tables will be used. Here we will focus only on achilles execution and skip running Heel.

```
connectionDetails <- createConnectionDetails(dbms = "postgresql",
                                             server = "localhost/synpuf",
                                             user = "cdm_user",
                                             password = "cdm_password")

achilles(connectionDetails = connectionDetails,
         cdmDatabaseSchema = "cdm",
         resultsDatabaseSchema = "results",
         vocabDatabaseSchema = "vocab",
         sourceName = "Synpuf",
```

```
          cdmVersion = 5.3,
          numThreads = 1,
          runHeel = FALSE)
```

5 Running Achilles: Multi-Threaded Mode

In multi-threaded mode, you need to specify `scratchDatabaseSchema` and use > 1 for `numThreads`. Here we will focus only on achilles execution and skip running Heel.

```
connectionDetails <- createConnectionDetails(dbms = "postgresql",
                                             server = "localhost/synpuf",
                                             user = "cdm_user",
                                             password = "cdm_password")

achilles(connectionDetails = connectionDetails,
          cdmDatabaseSchema = "cdm",
          resultsDatabaseSchema = "results",
          scratchDatabaseSchema = "scratch",
          vocabDatabaseSchema = "vocab",
          sourceName = "Synpuf",
          cdmVersion = 5.3,
          numThreads = 5,
          runHeel = FALSE)
```

6 Achilles Heel Parameters (Both Modes)

6.1 Staging Table Prefix

To keep the staging tables organized, the **achillesHeel** function will use a table prefix of "tmpheel" by default, but you can choose a different one using the `tempHeelPrefix` parameter. This is useful for database platforms like Oracle, which limit the length of table names.

6.2 Drop Scratch Tables

See the Post-Processing section to read about how to run this step separately

This parameter is only necessary if running in multi-threaded mode

The `dropScratchTables` parameter, if set to `TRUE` will drop all staging tables created during the execution of **achillesHeel** in multi-threaded mode.

6.3 Thresholds

The `ThresholdAgeWarning`, `ThresholdOutpatientVisitPerc`, and `ThresholdMinimalPtMeasDxRx` parameters can be used to configure DQ thresholds in **achillesHeel**.

- `ThresholdAgeWarning` refers to the maximum age to allow in the dataset; by default, this is 125 years of age.

- `ThresholdOutpatientVisitPerc` refers to the maximum percentage of outpatient visits allowed among all visits. This is by default set to 0.43.
- `ThresholdMinimalPtMeasDxRx` refers to the minimum percentage required of patients with at least 1 measurement, 1 condition, and 1 drug exposure. This is by default set to 20.5%.

7 Running Achilles Heel: Single-Threaded Mode

In single-threaded mode, there is no need to set a `scratchDatabaseSchema`, as temporary tables will be used.

```
connectionDetails <- createConnectionDetails(dbms = "postgresql",
                                             server = "localhost/synpuf",
                                             user = "cdm_user",
                                             password = "cdm_password")

achillesHeel(connectionDetails = connectionDetails,
             cdmDatabaseSchema = "cdm",
             resultsDatabaseSchema = "results",
             vocabDatabaseSchema = "vocab",
             cdmVersion = 5.3,
             numThreads = 1,
             outputFolder = "output")
```

8 Running Achilles Heel: Multi-Threaded Mode

In multi-threaded mode, you need to specify `scratchDatabaseSchema` and use > 1 for `numThreads`.

```
connectionDetails <- createConnectionDetails(dbms = "postgresql",
                                             server = "localhost/synpuf",
                                             user = "cdm_user",
                                             password = "cdm_password")
achillesHeel(connectionDetails = connectionDetails,
             cdmDatabaseSchema = "cdm",
             resultsDatabaseSchema = "results",
             vocabDatabaseSchema = "vocab",
             cdmVersion = 5.3,
             numThreads = 5,
             outputFolder = "output",
             scratchDatabaseSchema = "scratch")
```

9 Post-Processing

This section describes the usage of standalone functions for post-processing that can be invoked if you did not use them in the **achilles** function call.

9.1 Creating Indices

Not supported by Amazon Redshift or IBM Netezza; function will skip this step if using those platforms

To improve query performance of the Achilles results tables, run the **createIndices** function.

```
connectionDetails <- createConnectionDetails(dbms = "postgresql",
                                             server = "localhost/synpuf",
                                             user = "cdm_user",
                                             password = "cdm_password")

createIndices(connectionDetails = connectionDetails,
              resultsDatabaseSchema = "results",
              outputFolder = "output")
```

9.2 Dropping All Staging Tables (Multi-threaded only)

If the **achilles** execution has errors, or if you did not enable this step in the call to these functions, use the `dropAllScratchTables` function.

The `tableTypes` parameter can be used to specify which batch of staging tables to drop ("achilles", "heel"). The default value is to drop them all.

```
connectionDetails <- createConnectionDetails(dbms = "postgresql",
                                             server = "localhost/synpuf",
                                             user = "cdm_user",
                                             password = "cdm_password")

dropAllScratchTables(connectionDetails = connectionDetails,
                     scratchDatabaseSchema = "scratch", numThreads = 5)
```

9.3 Using AchillesWeb

AchillesWeb is a lightweight web application that can be used to consume the Achilles and Heel results graphically. It is no longer actively updated, as development priorities have shifted towards Atlas Data Sources, but AchillesWeb can still be utilized. To connect Achilles results to AchillesWeb, the Achilles results need to be exported to JSON files and the AchillesWeb JSON file needs to point to those JSON files.

Please refer to AchillesWeb for more information.

9.3.1 Exporting to JSON

The `exportToJson` function can export all of the Achilles results to JSON files that AchillesWeb can consume.

- The `compressIntoOnFile` parameter, if set to `TRUE`, will compress all of the files into one zip file for easier portability.
- The `reports` parameter can be used to export specific reports rather than all. Use the `showReportTypes` function to see all possible reports.

```
connectionDetails <- createConnectionDetails(dbms = "postgresql",
                                             server = "localhost/synpuf",
                                             user = "cdm_user",
                                             password = "cdm_password")

exportToJson(connectionDetails = connectionDetails,
```

```
              cdmDatabaseSchema = "cdm",
              resultsDatabaseSchema = "results",
              outputPath = "output",
              vocabDatabaseSchema = "vocab")
```

9.3.2 Adding Data Sources to AchillesWeb

AchillesWeb relies upon a JSON file to point to CDM Achilles result JSON files.

```
addDataSource(jsonFolderPath = "output",
               dataSourcePath = "achillesWeb")
```

10 Acknowledgments

Considerable work has been dedicated to provide the `Achilles` package.

```
citation("Achilles")
```

```
#>
#> To cite package 'Achilles' in publications use:
#>
#>   Frank DeFalco, Patrick Ryan, Martijn Schuemie, Vojtech Huser, Chris
#>   Knoll, Ajit Londhe and Taha Abdul-Basser (2019). Achilles: Creates
#>   Descriptive Statistics Summary for an Entire OMOP CDM Instance. R
#>   package version 1.6.7.
#>
#> A BibTeX entry for LaTeX users is
#>
#>   @Manual{,
#>     title = {Achilles: Creates Descriptive Statistics Summary for an Entire OMOP CDM Instance},
#>     author = {Frank DeFalco and Patrick Ryan and Martijn Schuemie and Vojtech Huser and Chris Knoll a
#>     year = {2019},
#>     note = {R package version 1.6.7},
#>   }
```