Package ‘CohortGenerator’

November 17, 2021

Type Package

Title An R Package for Cohort Generation Against the OMOP CDM

Version 0.2.0

Date 2021-11-17

Maintainer Anthony Sena <sena@ohdsi.org>

Description
An R package for that encapsulates the functions for generating cohorts against the OMOP CDM.

Depends DatabaseConnector (>= 4.0.0),
R (>= 3.6.0)

Imports digest,
ParallelLogger (>= 2.0.2),
readr (>= 1.4.0),
rlang,
RJSONIO,
SqlRender (>= 1.7.0),
methods,
dplyr,
stats

Suggests CirceR (>= 1.1.1),
Eunomia,
knitr,
testthat

Remotes ohdsi/CirceR,
ohdsi/Eunomia

License Apache License

URL https://ohdsi.github.io/CohortGenerator/, https://github.com/OHDSI/CohortGenerator

BugReports https://github.com/OHDSI/CohortGenerator/issues

RoxygenNote 7.1.1

Encoding UTF-8

Language en-US

R topics documented:

`computeChecksum` *Computes the checksum for a value*

Description

This is used as part of the incremental operations to hash a value to store in a record keeping file. This function leverages the md5 hash from the digest package

Usage

```
computeChecksum(val)
```

Arguments

`val`	The value to hash. It is converted to a character to perform the hash.

Value

Returns a string containing the checksum

`createCohortTable` *Create cohort table(s)*

Description

This function creates an empty cohort table. Optionally, additional empty tables are created to store statistics on the various inclusion criteria.

Usage

```
createCohortTable(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema,
  cohortTable = "cohort",
  createInclusionStatsTables = FALSE,
  resultsDatabaseSchema = cohortDatabaseSchema,
  cohortInclusionTable = paste0(cohortTable, "_inclusion"),
  cohortInclusionResultTable = paste0(cohortTable, "_inclusion_result"),
  cohortInclusionStatsTable = paste0(cohortTable, "_inclusion_stats"),
  cohortSummaryStatsTable = paste0(cohortTable, "_summary_stats"),
  cohortCensorStatsTable = paste0(cohortTable, "_censor_stats")
)
```

Arguments

`connectionDetails`
: An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.

`connection`
: An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if `connectionDetails` is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.

`cohortDatabaseSchema`
: Schema name where your cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`cohortTable`
: Name of the cohort table.

`createInclusionStatsTables`
: Create the four additional tables for storing inclusion rule statistics?

`resultsDatabaseSchema`
: Schema name where the statistics tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`cohortInclusionTable`
: Name of the inclusion table, one of the tables for storing inclusion rule statistics.

`cohortInclusionResultTable`
: Name of the inclusion result table, one of the tables for storing inclusion rule statistics.

`cohortInclusionStatsTable`
: Name of the inclusion stats table, one of the tables for storing inclusion rule statistics.

`cohortSummaryStatsTable`
: Name of the summary stats table, one of the tables for storing inclusion rule statistics.

`cohortCensorStatsTable`
: Name of the censor stats table, one of the tables for storing inclusion rule statistics.

createEmptyCohortSet *Create an empty cohort set*

Description

This function creates an empty cohort set data.frame for use with `generateCohortSet`.

Usage

```
createEmptyCohortSet()
```

Value

Returns an empty cohort set data.frame

generateCohort *Generates a cohort*

Description

This function is used by `generateCohortSet` to generate a cohort against the CDM.

Usage

```
generateCohort(
  cohortId = NULL,
  cohortSet,
  connection = NULL,
  connectionDetails = NULL,
  cdmDatabaseSchema,
  tempEmulationSchema,
  cohortDatabaseSchema,
  cohortTable,
  inclusionStatisticsFolder,
  incremental,
  recordKeepingFile
)
```

Arguments

`cohortId` The cohortId in the list of `cohortSet`

`cohortSet` The `cohortSet` argument must be a data frame with the following columns:

cohortId The unique integer identifier of the cohort

cohortFullName The cohort's full name

sql The OHDSI-SQL used to generate the cohort

json The json column must represent a Circe cohort definition. This field is only required when you would like to generate a cohort that includes inclusion statistics since the names of the inclusion rules are parsed from this JSON property.

`connection` An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if `connectionDetails` is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.

`connectionDetails`
An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.

`cdmDatabaseSchema`
Schema name where your patient-level data in OMOP CDM format resides. Note that for SQL Server, this should include both the database and schema name, for example 'cdm_data.dbo'.

`tempEmulationSchema`
Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.

`cohortDatabaseSchema`
Schema name where your cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`cohortTable` Name of the cohort table.

`inclusionStatisticsFolder`
The folder where the inclusion rule statistics are stored. Can be left NULL if you do not wish to export the inclusion rule statistics

`incremental` Create only cohorts that haven't been created before?

`recordKeepingFile`
If `incremental` = `TRUE`, this file will contain information on cohorts already generated

---

`generateCohortSet` *Generate a set of cohorts*

Description

This function generates a set of cohorts in the cohort table and where specified the inclusion rule statistics are computed and stored in the `inclusionStatisticsFolder`.

Usage

```
generateCohortSet(
  connectionDetails = NULL,
  connection = NULL,
  numThreads = 1,
  cdmDatabaseSchema,
  tempEmulationSchema = NULL,
  cohortDatabaseSchema = cdmDatabaseSchema,
  cohortTable = "cohort",
  cohortSet = NULL,
  inclusionStatisticsFolder = NULL,
  createCohortTable = FALSE,
  incremental = FALSE,
  incrementalFolder = NULL
)
```

Arguments

`connectionDetails`
: An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.

`connection`
: An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if `connectionDetails` is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.

`numThreads`
: Specify the number of threads for cohort generation. Currently this only supports single threaded operations.

`cdmDatabaseSchema`
: Schema name where your patient-level data in OMOP CDM format resides. Note that for SQL Server, this should include both the database and schema name, for example 'cdm_data.dbo'.

`tempEmulationSchema`
: Some database platforms like Oracle and Impala do not truly support temp tables. To emulate temp tables, provide a schema with write privileges where temp tables can be created.

`cohortDatabaseSchema`
: Schema name where your cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`cohortTable`
: Name of the cohort table.

`cohortSet`
: The `cohortSet` argument must be a data frame with the following columns:

cohortId The unique integer identifier of the cohort

cohortFullName The cohort's full name

sql The OHDSI-SQL used to generate the cohort

json The json column must represent a Circe cohort definition. This field is only required when you would like to generate a cohort that includes inclusion statistics since the names of the inclusion rules are parsed from this JSON property.

`inclusionStatisticsFolder`
: The folder where the inclusion rule statistics are stored. Can be left NULL if you do not wish to export the inclusion rule statistics

`createCohortTable`
: Create the cohort table? If `incremental` = `TRUE` and the table already exists this will be skipped.

`incremental`
: Create only cohorts that haven't been created before?

`incrementalFolder`
: If `incremental` = `TRUE`, specify a folder where records are kept of which definition has been executed.

getCohortCounts *Count the cohort(s)*

Description

Computes the subject and entry count per cohort

Usage

```
getCohortCounts(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema,
  cohortTable = "cohort",
  cohortIds = c()
)
```

Arguments

`connectionDetails`
An object of type `connectionDetails` as created using the `createConnectionDetails` function in the DatabaseConnector package. Can be left NULL if `connection` is provided.

`connection`
An object of type `connection` as created using the `connect` function in the DatabaseConnector package. Can be left NULL if `connectionDetails` is provided, in which case a new connection will be opened at the start of the function, and closed when the function finishes.

`cohortDatabaseSchema`
Schema name where your cohort table resides. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`cohortTable`
Name of the cohort table.

`cohortIds`
The cohort Id(s) used to reference the cohort in the cohort table. If left empty, all cohorts in the table will be included.

Value

A data frame with cohort counts

getRequiredTasks *Get a list of tasks required when running in incremental mode*

Description

This function will attempt to check the `recordKeepingFile` to determine if a list of operations have completed by comparing the keys passed into the function with the checksum supplied

Usage

```
getRequiredTasks(..., checksum, recordKeepingFile)
```

Arguments

`...` Parameter values used to identify the key in the incremental record keeping file

`checksum` The checksum representing the operation to check

`recordKeepingFile` A file path to a CSV file containing the record keeping information.

Value

Returns a list of outstanding tasks based on inspecting the full contents of the record keeping file

`isTaskRequired` *Is a task required when running in incremental mode*

Description

This function will attempt to check the `recordKeepingFile` to determine if an individual operation has completed by comparing the keys passed into the function with the checksum supplied

Usage

```
isTaskRequired(..., checksum, recordKeepingFile, verbose = TRUE)
```

Arguments

`...` Parameter values used to identify the key in the incremental record keeping file

`checksum` The checksum representing the operation to check

`recordKeepingFile` A file path to a CSV file containing the record keeping information.

`verbose` When TRUE, this function will output if a particular operation has completed based on inspecting the recordKeepingFile.

Value

Returns TRUE if the operation has completed according to the contents of the record keeping file.

`recordTasksDone` *Record a task as complete*

Description

This function will record a task as completed in the `recordKeepingFile`

Usage

```
recordTasksDone(..., checksum, recordKeepingFile, incremental = TRUE)
```

Arguments

`...` Parameter values used to identify the key in the incremental record keeping file

`checksum` The checksum representing the operation to check

`recordKeepingFile` A file path to a CSV file containing the record keeping information.

`incremental` When TRUE, this function will record tasks otherwise it will return without attempting to perform any action

`saveIncremental` *Used in incremental mode to save values to a file*

Description

When running in incremental mode, we may need to update results in a CSV file. This function will replace the `data` in `fileName` based on the key parameters

Usage

```
saveIncremental(data, fileName, ...)
```

Arguments

`data` The data to record in the file

`fileName` A CSV holding results in the same structure as the data parameter

`...` Parameter values used to identify the key in the results file

`sqlContainsInclusionRuleStats` *Detects if the SQL indicate to compute inclusion rule statistics*

Description

This function takes as a parameter a SQL script used to generate a cohort and performs a string search for tokens that indicate to generate the inclusion statistics. This SQL is usually generated by circe-be.

This function also assumes that the SQL passed into the function has not been translated to a specific SQL dialect.

Usage

```
sqlContainsInclusionRuleStats(sql)
```

Arguments

`sql` A string containing the SQL used to generate the cohort. This code assumes that the SQL has not been rendered using SqlRender in order to detect tokens that indicate the generation of inclusion rule statistics in addition to the cohort.

Index