Package 'CohortGenerator'

August 16, 2021

Type Package

Title An R Package for Cohort Generation Against the OMOP CDM

Version 0.1.1

Date 2021-08-10

Maintainer Anthony Sena <sena@ohdsi.org>

Description An R package for that encapsulates the functions for generating cohorts against the OMOP CDM. This package allow for the use of SQL generated using CIRCE-compliant JSON or custom SQL.

Depends DatabaseConnector (>= 4.0.0),
CirceR (>= 1.1.1),
R (>= 3.5.0)

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

Suggests Eunomia,
knitr,
testthat

Remotes ohdsi/CirceR

License Apache License

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

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

RoxygenNote 7.1.1

Encoding UTF-8

Language en-US

R topics documented:

computeChecksum 2
createCirceExpressionFromFile 2
createCohortTable 3
createEmptyCohortSet 4
generateCohort 4
generateCohortSet 5
getRequiredTasks 7
isTaskRequired 7
loadRenderTranslateSql 8
readCirceExpressionJsonFile 9
recordTasksDone 9
saveIncremental 10
sqlContainsInclusionRuleStats 10

Index **11**

`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

`createCirceExpressionFromFile`
Create the Circe cohort expression from a JSON file

Description

This function constructs a Circe cohort expression from a JSON file for use with other CirceR functions.

Usage

```
createCirceExpressionFromFile(filePath)
```

Arguments

filePath	The file path containing the Circe JSON file

---

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.

`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.

---

loadRenderTranslateSql

Load, render, and translate a SQL file in this package.

Description

This helper function is used in place of using `SqlRender::loadRenderTranslateSql` otherwise unit tests will not function properly.

NOTE: This function does not support dialect-specific SQL translation at this time.

Usage

```
loadRenderTranslateSql(
  sqlFilename,
  dbms = "sql server",
  ...,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  warnOnMissingParameters = TRUE
)
```

Arguments

sqlFilename	The source SQL file
dbms	The target dialect. Currently 'sql server', 'oracle', 'postgres', and 'redshift' are supported
...	Parameter values used for `render`
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.
warnOnMissingParameters	Should a warning be raised when parameters provided to this function do not appear in the parameterized SQL that is being rendered? By default, this is TRUE.

Value

Returns a string containing the rendered SQL.

readCirceExpressionJsonFile *Read a JSON file for use with CirceR*

Description

This function wraps the default readlines call for calling CirceR.

Usage

```
readCirceExpressionJsonFile(filePath)
```

Arguments

filePath	The file path containing the Circe JSON 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

computeChecksum, 2
connect, *3*, *5, 6*
createCirceExpressionFromFile, 2
createCohortTable, 3
createConnectionDetails, *3*, *5, 6*
createEmptyCohortSet, 4

generateCohort, 4
generateCohortSet, 5
getRequiredTasks, 7

isTaskRequired, 7

loadRenderTranslateSql, 8

readCirceExpressionJsonFile, 9
recordTasksDone, 9

saveIncremental, 10
sqlContainsInclusionRuleStats, 10