Package 'CohortAlgebra'

May 31, 2023

Type Package

Title Cohort Algebra to create new cohort(s) from existing cohorts

Version 0.0.1

Date 2023-01-11

Maintainer Gowtham Rao <rao@ohdsi.org>

Description An R package that creates new cohort(s) from previously instantiated cohorts.

Depends DatabaseConnector (>= 5.0.0),
R (>= 4.1.0)

Imports checkmate,
clock,
dplyr,
lifecycle,
ParallelLogger,
rlang,
SqlRender

Suggests Eunomia,
remotes,
rmarkdown,
knitr,
testthat,
withr

Remotes ohdsi/Eunomia,
ohdsi/ParallelLogger

License Apache License

RoxygenNote 7.2.3

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Encoding UTF-8

Language en-US

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

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

R topics documented:

appendCohortTables *Append cohort data from multiple cohort tables(s)*

Description

Append cohort data from multiple cohort tables.

[Stable]

Usage

```
appendCohortTables(
  connectionDetails = NULL,
  connection = NULL,
  sourceTables,
  targetCohortDatabaseSchema = NULL,
  targetCohortTable,
  isTempTable = FALSE,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema")
)
```

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.

`sourceTables`
A data.frame object with the columns sourceCohortDatabaseSchema, sourceCohortTableName.

`targetCohortDatabaseSchema`
Schema name where your target cohort tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`targetCohortTable`
The name of the target cohort table.

`isTempTable` Is the output a temp table. If yes, a new temp table is created. This will required an active connection. Any old temp table is dropped and replaced.

`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.

Examples

```
## Not run:
appendCohortTables(
  connectionDetails = Eunomia::getEunomiaConnectionDetails(),
 sourceTables = dplyr::tibble(sourceCohortDatabaseSchema = "main", sourceCohortTableName = "cohort"),
  targetCohortDatabaseSchema = "main",
  targetCohortTable = "target"
)

## End(Not run)
```

`copyCohorts` *Copy cohorts from one table to another*

Description

Copy cohorts from one table to another table. If the new cohort table has any cohort id that matches the cohort id being copied, an error will be displayed.

[Stable]

Usage

```
copyCohorts(
  connectionDetails = NULL,
  connection = NULL,
  oldToNewCohortId,
  sourceCohortDatabaseSchema = NULL,
  targetCohortDatabaseSchema = sourceCohortDatabaseSchema,
  sourceCohortTable,
  targetCohortTable,
  purgeConflicts = FALSE,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema")
)
```

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.

`oldToNewCohortId`
A data.frame object with two columns. oldCohortId and newCohortId. Both should be integers. The oldCohortId are the cohorts that are the input cohorts that need to be transformed. The newCohortId are the cohortIds of the corresponding output after transformation. If the oldCohortId = newCohortId then the data corresponding to oldCohortId will be replaced by the data from the newCohortId.

`sourceCohortDatabaseSchema`
The database schema of the source cohort table.

`targetCohortDatabaseSchema`
The database schema of the source cohort table.

`sourceCohortTable`
The name of the source cohort table.

`targetCohortTable`
The name of the target cohort table.

`purgeConflicts` If there are conflicts in the target cohort table i.e. the target cohort table already has records with newCohortId, do you want to purge and replace them with transformed. By default - it will not be replaced, and an error message is thrown.

`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.

Examples

```
## Not run:
CohortAlgebra::copyCohorts(
  connection = connection,
  sourceCohortDatabaseSchema = cohortDatabaseSchema,
  targetCohortDatabaseSchema = cohortDatabaseSchema,
  sourceCohortTable = tableName,
  targetCohortTable = tableName,
  purgeConflicts = TRUE
)

## End(Not run)
```

`copyCohortsToTempTable`

Copy cohorts to temp table

Description

Copy cohorts to temp table. This function is not exported.

[Stable]

Usage

```
copyCohortsToTempTable(
  connection = NULL,
  oldToNewCohortId,
  sourceCohortDatabaseSchema = NULL,
  sourceCohortTable,
  targetCohortTable = "#cohort_rows",
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema")
)
```

Arguments

`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.

`oldToNewCohortId`
A data.frame object with two columns. oldCohortId and newCohortId. Both should be integers. The oldCohortId are the cohorts that are the input cohorts that need to be transformed. The newCohortId are the cohortIds of the corresponding output after transformation. If the oldCohortId = newCohortId then the data corresponding to oldCohortId will be replaced by the data from the newCohortId.

`sourceCohortDatabaseSchema`
The database schema of the source cohort table.

`sourceCohortTable`
The name of the source cohort table.

`targetCohortTable`
A temp table to copy the cohorts from the source table.

`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.

`deleteCohort` *Delete cohort*

Description

Delete all records for a given set of cohorts from the cohort table. Edit privileges to the cohort table is required.

[Stable]

Usage

```
deleteCohort(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema,
  cohortTable = "cohort",
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  cohortIds
)
```

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 tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

cohortTable
: The name of the cohort table.

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.

cohortIds
: A vector of one or more Cohort Ids.

`eraFyCohorts` *Era-fy cohort(s)*

Description

Given a table with cohort_definition_id, subject_id, cohort_start_date, cohort_end_date execute era logic. This will delete and replace the original rows with the cohort_definition_id(s). edit privileges to the cohort table is required.

[Stable]

Usage

```
eraFyCohorts(
  connectionDetails = NULL,
  connection = NULL,
  sourceCohortDatabaseSchema = NULL,
  sourceCohortTable = "cohort",
  targetCohortDatabaseSchema = NULL,
  targetCohortTable,
  oldCohortIds,
  newCohortId,
  eraconstructorpad = 0,
  cdmDatabaseSchema = NULL,
  purgeConflicts = FALSE,
  isTempTable = FALSE,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema")
)
```

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.

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

`sourceCohortTable`
: The name of the source cohort table.

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

`targetCohortTable`
: The name of the target cohort table.

`oldCohortIds`
: An array of 1 or more integer id representing the cohort id of the cohort on which the function will be applied.

`newCohortId`
: The cohort id of the output cohort.

`eraconstructorpad`
: Optional value to pad cohort era construction logic. Default = 0. i.e. no padding.

`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'.

`purgeConflicts`
: If there are conflicts in the target cohort table i.e. the target cohort table already has records with newCohortId, do you want to purge and replace them with transformed. By default - it will not be replaced, and an error message is thrown.

`isTempTable`
: Is the output a temp table. If yes, a new temp table is created. This will required an active connection. Any old temp table is dropped and replaced.

`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.

`getCohortIdsInCohortTable`

Get cohort ids in table

Description

Get cohort ids in table

[Stable]

Usage

```
getCohortIdsInCohortTable(
  connection = NULL,
  cohortDatabaseSchema = NULL,
  cohortTable,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema")
)
```

Arguments

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

`cohortTable` The name of the cohort table.

`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.

`intersectCohorts` *Intersect cohort(s)*

Description

Find the common cohort period for persons present in all the cohorts. Note: if subject is not found in any of the cohorts, then they will not be in the final cohort.

[Stable]

Usage

```
intersectCohorts(
  connectionDetails = NULL,
  connection = NULL,
  sourceCohortDatabaseSchema = NULL,
  sourceCohortTable,
  targetCohortDatabaseSchema = NULL,
  targetCohortTable,
  cohortIds,
  newCohortId,
  purgeConflicts = FALSE,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema")
)
```

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.

`sourceCohortDatabaseSchema`
Schema name where your source cohort tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`sourceCohortTable`
The name of the source cohort table.

`targetCohortDatabaseSchema`
Schema name where your target cohort tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`targetCohortTable`
The name of the target cohort table.

`cohortIds`
A vector of one or more Cohort Ids.

`newCohortId`
The cohort id of the output cohort.

`purgeConflicts`
If there are conflicts in the target cohort table i.e. the target cohort table already has records with newCohortId, do you want to purge and replace them with transformed. By default - it will not be replaced, and an error message is thrown.

`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.

Examples

```
## Not run:
intersectCohorts(
  connectionDetails = Eunomia::getEunomiaConnectionDetails(),
  sourceCohortDatabaseSchema = "main",
  sourceCohortTable = "cohort",
  cohortIds = c(1, 2, 3),
  newCohortId = 9,
  purgeConflicts = TRUE
)

## End(Not run)
```

`minusCohorts` *Minus cohort(s)*

Description

Given two cohorts, substract (minus) the dates from the first cohort, the dates the subject also had on the second cohort.

[Stable]

Usage

```
minusCohorts(
  connectionDetails = NULL,
  connection = NULL,
  sourceCohortDatabaseSchema = NULL,
  sourceCohortTable = "cohort",
  targetCohortDatabaseSchema = sourceCohortDatabaseSchema,
  targetCohortTable = sourceCohortTable,
  firstCohortId,
  secondCohortId,
  newCohortId,
  purgeConflicts = FALSE,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema")
)
```

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.
sourceCohortDatabaseSchema	Schema name where your source cohort tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.
sourceCohortTable	The name of the source cohort table.
targetCohortDatabaseSchema	Schema name where your target cohort tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.
targetCohortTable	The name of the target cohort table.
firstCohortId	The cohort id of the cohort from which to subtract.
secondCohortId	The cohort id of the cohort that is used to subtract.
newCohortId	The cohort id of the output cohort.
purgeConflicts	If there are conflicts in the target cohort table i.e. the target cohort table already has records with newCohortId, do you want to purge and replace them with transformed. By default - it will not be replaced, and an error message is thrown.
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.

Examples

```
## Not run:
minusCohorts(
  connectionDetails = Eunomia::getEunomiaConnectionDetails(),
  sourceCohortDatabaseSchema = "main",
  sourceCohortTable = "cohort",
  firstCohortId = 1,
  secondCohortId = 2,
  newCohortId = 9,
  purgeConflicts = TRUE
)

## End(Not run)
```

removeOverlappingSubjects

Remove subjects in cohort that overlap with another cohort

Description

Remove subjects in cohort that overlap with another cohort. Given a Cohort A, check if the records of subjects in cohort A overlaps with records for the same subject in cohort B. If there is overlap then remove all records of that subject from Cohort A. Overlap is defined as b.cohort_end_date >= a.cohort_start_date AND b.cohort_start_date <= a.cohort_end_date. The overlap logic maybe offset by using a startDayOffSet (applied on cohort A's cohort_start_date) and endDayOffSet (applied on Cohort A's cohort_end_date). If while applying offset, the window becomes such that (a.cohort_start_date + startDayOffSet) > (a.cohort_end_date + endDayOffset) that record is ignored and thus deleted.

[Experimental]

Usage

```
removeOverlappingSubjects(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema,
  cohortId,
  newCohortId,
  cohortsWithSubjectsToRemove,
  offsetCohortStartDate = -99999,
  offsetCohortEndDate = 99999,
  restrictSecondCohortStartBeforeFirstCohortStart = FALSE,
  restrictSecondCohortStartAfterFirstCohortStart = FALSE,
  cohortTable = "cohort",
  purgeConflicts = FALSE,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema")
)
```

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 tables reside. Note that for SQL Server, this should include both the database and schema name, for example 'scratch.dbo'.

`cohortId` The cohort id of the cohort whose subjects will be removed.

`newCohortId` The cohort id of the output cohort.

`cohortsWithSubjectsToRemove` An array of one or more cohorts with subjects to remove from given cohorts.

`offsetCohortStartDate` (Default = 0) If you want to offset cohort start date, please provide a integer number.

`offsetCohortEndDate` (Default = 0) If you want to offset cohort start date, please provide a integer number.

`restrictSecondCohortStartBeforeFirstCohortStart` (Default = FALSE) If TRUE, then the secondCohort's cohort_start_date should be < firstCohort's cohort_start_date.

`restrictSecondCohortStartAfterFirstCohortStart` (Default = FALSE) If TRUE, then the secondCohort's cohort_start_date should be > firstCohort's cohort_start_date.

`cohortTable` The name of the cohort table.

`purgeConflicts` If there are conflicts in the target cohort table i.e. the target cohort table already has records with newCohortId, do you want to purge and replace them with transformed. By default - it will not be replaced, and an error message is thrown.

`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.

Examples

```
## Not run:
removeOverlappingSubjects(
  connection = connection,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cohortId = 1,
  newCohortId = 9,
  cohortsWithSubjectsToRemove = c(3),
  purgeConflicts = FALSE,
  cohortTable = tableName
)

## End(Not run)
```

unionCohorts *Union cohort(s)*

Description

Given a specified array of cohortIds in a cohort table, perform cohort union operator to create new cohorts.

[Stable]

Usage

```
unionCohorts(
  connectionDetails = NULL,
  connection = NULL,
  sourceCohortDatabaseSchema = NULL,
  sourceCohortTable,
  targetCohortDatabaseSchema = NULL,
  targetCohortTable,
  oldToNewCohortId,
  isTempTable = FALSE,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  purgeConflicts = FALSE
)
```

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.

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

`sourceCohortTable`
: The name of the source cohort table.

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

`targetCohortTable`
: The name of the target cohort table.

`oldToNewCohortId`
: A data.frame object with two columns. oldCohortId and newCohortId. Both should be integers. The oldCohortId are the cohorts that are the input cohorts that need to be transformed. The newCohortId are the cohortIds of the corresponding output after transformation. If the oldCohortId = newCohortId then the data corresponding to oldCohortId will be replaced by the data from the newCohortId.

`isTempTable` Is the output a temp table. If yes, a new temp table is created. This will required an active connection. Any old temp table is dropped and replaced.

`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.

`purgeConflicts` If there are conflicts in the target cohort table i.e. the target cohort table already has records with newCohortId, do you want to purge and replace them with transformed. By default - it will not be replaced, and an error message is thrown.

Examples

```
## Not run:
unionCohorts(
  connectionDetails = Eunomia::getEunomiaConnectionDetails(),
  sourceDatabaseSchema = "main",
  sourceCohortTable = "cohort",
  oldToNewCohortId = dplyr::tibble(oldCohortId = c(1, 2), newCohortId = 4),
  purgeConflicts = TRUE
)

## End(Not run)
```

Index