Package 'CohortAlgebra'

August 9, 2022

Type Package

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

Version 0.3.0

Date 2022-08-07

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,
ParallelLogger,
rlang,
SqlRender

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

Remotes ohdsi/Eunomia,
ohdsi/ParallelLogger

License Apache License

RoxygenNote 7.2.1

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:

`copyCohortsToTempTable` *Get cohort ids in table*

Description

Get cohort ids in table. This function is not exported.

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.

`deleteCohortRecords` *Delete cohort records.*

Description

Delete all records from cohort table with the given cohort id. Edit privileges to the cohort table is required.

Usage

```
deleteCohortRecords(
  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.

Usage

```
eraFyCohorts(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema = NULL,
  cohortTable = "cohort",
  oldToNewCohortId,
  eraconstructorpad = 0,
  cdmDatabaseSchema = NULL,
  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.

`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.

`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.

`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'. cdmDataschema is required when eraConstructorPad is > 0. eraConstructorPad is optional.

`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.

Value

NULL s

`getCohortIdsInCohortTable` *Get cohort ids in table*

Description

Get cohort ids in table

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.

Usage

```
intersectCohorts(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema = NULL,
  cohortTable = "cohort",
  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.
`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.
`cohortIds`	A vector of one or more Cohort Ids.
`newCohortId`	The cohort id of the result 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.

Value

NULL

intersectCohorts(connectionDetails = Eunomia::getEunomiaConnectionDetails(), cohortDatabaseSchema = "main", cohortTable = "cohort", cohortIds = c(1, 2, 3), newCohortId = 9, purgeConflicts = TRUE)

`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.

Usage

```
minusCohorts(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema = NULL,
  cohortTable = "cohort",
  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.
`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.
`firstCohortId`	The cohort id of the cohort from which to substract.
`secondCohortId`	The cohort id of the cohort that is used to substract.
`newCohortId`	The cohort id of the result 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

```
minusCohorts(
  connectionDetails = Eunomia::getEunomiaConnectionDetails(),
  cohortDatabaseSchema = "main",
  cohortTable = "cohort",
  firstCohortId = 1,
  secondCohortId = 2,
  newCohortId = 9,
  purgeConflicts = TRUE
)
```

`modifyCohort` *Modify cohort*

Description

Modify cohort by censoring, padding, limiting cohorts periods. Censoring: Provide a date for right, left, both censoring. All cohorts will be truncated to the given date. Pad days: Add days to either cohort start or cohort end dates. Maybe negative numbers. Final cohort will not be outside the persons observation period. Limit cohort periods: Filter the cohorts to a given date range of cohort start, or cohort end or both.

Usage

```
modifyCohort(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema = NULL,
  cohortTable = "cohort",
  oldCohortId,
  newCohortId = oldCohortId,
  cohortStartCensorDate = NULL,
  cohortEndCensorDate = NULL,
  cohortStartFilterRange = NULL,
  cohortEndFilterRange = NULL,
  cohortStartPadDays = NULL,
  cohortEndPadDays = NULL,
  cdmDatabaseSchema = NULL,
  tempEmulationSchema = getOption("sqlRenderTempEmulationSchema"),
  purgeConflicts = TRUE
)
```

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.

`oldCohortId` The cohort id of the cohort that needs to be modified.

`newCohortId` The cohort id of the result cohort.

`cohortStartCensorDate` the minimum date for the cohort. All rows with cohort start date before this date will be censored to given date.

`cohortEndCensorDate` the maximum date for the cohort. All rows with cohort end date after this date will be censored to given date.

`cohortStartFilterRange` A range of dates representing minimum to maximum to filter the cohort by its cohort start date e.g c(as.Date('1999-01-01'), as.Date('1999-12-31'))

`cohortEndFilterRange` A range of dates representing minimum to maximum to filter the cohort by its cohort end date e.g c(as.Date('1999-01-01'), as.Date('1999-12-31'))

`cohortStartPadDays` An integer value to pad the cohort start date. Default is 0 - no padding. The final cohort will have no days outside the observation period dates.

`cohortEndPadDays` An integer value to pad the cohort end date. Default is 0 - no padding. The final cohort will have no days outside the observation period dates.

`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'. cdmDataschema is required when eraConstructorPad is > 0. eraConstructorPad is optional.

`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

```
CohortAlgebra:::modifyCohort(
 connection = connection,
 cohortDatabaseSchema = cohortDatabaseSchema,
 cohortTable = tableName,
 oldCohortId = 3,
 newCohortId = 2,
 cohortEndFilterRange = c(as.Date("2010-01-01"), as.Date("2010-01-09")),
 purgeConflicts = TRUE
```

```
)
```

`removeSubjectsFromCohorts` *Remove subjects from cohort(s).*

Description

Remove subjects from a given array of cohort(s) who are present in any of another array of cohort(s).

Usage

```
removeSubjectsFromCohorts(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema,
  oldToNewCohortId,
  cohortsWithSubjectsToRemove,
  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'.

`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.

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

`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

```
removeSubjectsFromCohorts(
  connection = connection,
  cohortDatabaseSchema = cohortDatabaseSchema,
  oldToNewCohortId = dplyr::tibble(oldCohortId = 1, newCohortId = 6),
  cohortsWithSubjectsToRemove = c(3),
  purgeConflicts = FALSE,
  cohortTable = tableName
)
```

---

unionCohorts *Union cohort(s)*

Description

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

Usage

```
unionCohorts(
  connectionDetails = NULL,
  connection = NULL,
  cohortDatabaseSchema = NULL,
  cohortTable = "cohort",
  oldToNewCohortId,
  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.

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.

`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.

`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

```
unionCohorts(
  connectionDetails = Eunomia::getEunomiaConnectionDetails(),
  cohortDatabaseSchema = "main",
  cohortTable = "cohort",
  oldToNewCohortId = dplyr::tibble(
    oldCohortId = c(1, 2, 3),
    newCohortId = c(9, 9, 9)
  ),
  purgeConflicts = TRUE
)
```

Index