Package 'CohortConstructor'

July 31, 2024

Title Build and Manipulate Study Cohorts Using a Common Data Model

Version 0.2.2

Description Create and manipulate study cohorts in data mapped to the Observational Medical Outcomes Partnership Common Data Model.

License Apache License (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

Imports CDMConnector (>= 1.3.1), checkmate, cli, dbplyr (>= 2.5.0), dplyr, glue, magrittr, omopgenerics (>= 0.2.1), PatientProfiles (>= 1.1.0), purrr, rlang, tidyr, utils

Suggests DBI, CodelistGenerator, DrugUtilisation, duckdb, knitr, rmarkdown, testthat (>= 3.0.0), tibble, stringr, IncidencePrevalence, omock (>= 0.2.0), covr, RPostgres, odbc

Config/testthat/edition 3

Config/testthat/parallel true

VignetteBuilder knitr

Depends R (>= 4.1)

URL https://ohdsi.github.io/CohortConstructor/

NeedsCompilation no

Author Edward Burn [aut, cre] (<https://orcid.org/0000-0002-9286-1128>), Marti Catala [aut] (<https://orcid.org/0000-0003-3308-9905>), Nuria Mercade-Besora [aut] (<https://orcid.org/0009-0006-7948-3747>), Marta Alcalde-Herraiz [aut] (<https://orcid.org/0009-0002-4405-1814>), Mike Du [aut] (<https://orcid.org/0000-0002-9517-8834>), Yuchen Guo [aut] (<https://orcid.org/0000-0002-0847-4855>), Xihang Chen [aut] (<https://orcid.org/0009-0001-8112-8959>), Kim Lopez [aut] (<https://orcid.org/0000-0002-8462-8668>)

Maintainer Edward Burn <edward.burn@ndorms.ox.ac.uk>

Repository CRAN

Date/Publication 2024-07-31 09:50:02 UTC

Contents

`collapseCohorts` *Collapse cohort entries using a certain gap to concatenate records.*

Description

`collapseCohorts()` concatenates cohort records, allowing for some number of days between one finishing and the next starting.

Usage

```
collapseCohorts(cohort, cohortId = NULL, gap = 0, name = tableName(cohort))
```

Arguments

cohort	A cohort table
cohortId	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
gap	Number of days to use when merging cohort entries.
name	Name of the cohort table.

Value

A cohort table

---

conceptCohort *Create cohorts based on a concept set*

Description

`conceptCohort()` creates a cohort table from patient records from the clinical tables in the OMOP CDM.

The following tables are currently supported for creating concept cohorts:

- condition_occurrence
- device_exposure
- drug_exposure
- measurement
- observation
- procedure_occurrence
- visit_occurrence

Cohort duration is based on record start and end (e.g. condition_start_date and condition_end_date for records coming from the condition_occurrence tables). So that the resulting table satisfies the requirements of an OMOP CDM cohort table:

- Overlapping records are collapsed into a single cohort entry.
- If a record starts outside of an observation period it will be silently ignored.
- If a record ends outside of an observation period it will be trimmed so as to end at the preceding observation period end date.

Usage

```
conceptCohort(cdm, conceptSet, name)
```

Arguments

cdm	A cdm reference.
conceptSet	A conceptSet, which can either be a codelist or a conceptSetExpression.
name	Name of the cohort in the cdm object.

Value

A cohort table

Examples

```
library(CohortConstructor)

cdm <- mockCohortConstructor(conditionOccurrence = TRUE)

cohort <- conceptCohort(cdm = cdm, conceptSet = list(a = 1), name = "cohort")

cohort |> attrition()
```

---

demographicsCohort *Create cohorts based on patient demographics*

Description

demographicsCohort() creates a cohort table based on patient characteristics. If and when an individual satisfies all the criteria they enter the cohort. When they stop satisfying any of the criteria their cohort entry ends.

Usage

```
demographicsCohort(
  cdm,
  name,
  ageRange = NULL,
  sex = NULL,
  minPriorObservation = NULL,
  minFutureObservation = NULL
)
```

Arguments

cdm	A cdm reference.
name	Name of the new cohort table
ageRange	A list of vectors specifying minimum and maximum age.
sex	Can be "Both", "Male" or "Female".

`minPriorObservation`
: A minimum number of prior observation days in the database.

`minFutureObservation`
: A minimum number of future observation days in the database.

Value

A cohort table

Examples

```
library(CohortConstructor)

cdm <- mockCohortConstructor()

cohort <-  cdm |> demographicsCohort(name = "cohort3", ageRange = c(18,40), sex = "Male")

attrition(cohort)
```

---

entryAtFirstDate *Update cohort start date to be the first date from of a set of column dates*

Description

`entryAtFirstDate()` resets cohort start date based on a set of specified column dates. The first date that occurs is chosen.

Usage

```
entryAtFirstDate(
  cohort,
  dateColumns,
  cohortId = NULL,
  returnReason = TRUE,
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`dateColumns`	Date columns in the cohort table to consider.
`cohortId`	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
`returnReason`	If TRUE it will return a column stating which column in `dateColumns` is used as a new cohort end date.
`name`	Name of the new cohort with the restriction.

Value

The cohort table.

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor(tables = list(
"cohort" = dplyr::tibble(
  cohort_definition_id = 1,
  subject_id = c(1, 2, 3, 4),
  cohort_start_date = as.Date(c("2000-06-03", "2000-01-01", "2015-01-15", "2000-12-09")),
  cohort_end_date = as.Date(c("2001-09-01", "2001-01-12", "2015-02-15", "2002-12-09")),
  date_1 = as.Date(c("2001-08-01", "2001-01-01", "2015-01-15", "2002-12-09")),
  date_2 = as.Date(c("2001-08-01", NA, "2015-02-14", "2002-12-09"))
)
))
cdm$cohort |> entryAtLastDate(dateColumns = c("date_1", "date_2"))
```

`entryAtLastDate` *Set cohort start date to the last of a set of column dates*

Description

Set cohort start date to the last of a set of column dates

Usage

```
entryAtLastDate(
  cohort,
  dateColumns,
  cohortId = NULL,
  returnReason = TRUE,
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`dateColumns`	description
`cohortId`	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
`returnReason`	If TRUE it will return a column stating which column in `dateColumns` is used as a new cohort end date. description
`name`	Name of the new cohort with the restriction.

Value

The cohort table.

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor(tables = list(
"cohort" = dplyr::tibble(
  cohort_definition_id = 1,
  subject_id = c(1, 2, 3, 4),
  cohort_start_date = as.Date(c("2000-06-03", "2000-01-01", "2015-01-15", "2000-12-09")),
  cohort_end_date = as.Date(c("2001-09-01", "2001-01-12", "2015-02-15", "2002-12-09")),
  date_1 = as.Date(c("2001-08-01", "2001-01-01", "2015-01-15", "2002-12-09")),
  date_2 = as.Date(c("2001-08-01", NA, "2015-02-14", "2002-12-09"))
)
))
cdm$cohort |> entryAtLastDate(dateColumns = c("date_1", "date_2"))
```

---

exitAtDeath *Set cohort end date to death date*

Description

This functions changes cohort end date to subject's death date. In the case were this generates overlapping records in the cohort, those overlapping entries will be merged.

Usage

```
exitAtDeath(
  cohort,
  cohortId = NULL,
  requireDeath = FALSE,
  name = tableName(cohort)
)
```

Arguments

cohort	A cohort table in a cdm reference.
cohortId	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
requireDeath	If TRUE, subjects without a death record will be dropped, while if FALSE their end date will be left as is.
name	Name of the new cohort with the restriction.

Value

The cohort table.

Examples

```
library(PatientProfiles)
library(CohortConstructor)
cdm <- mockPatientProfiles()
cdm$cohort1 |> exitAtDeath()
```

`exitAtFirstDate` *Set cohort end date to the first of a set of column dates*

Description

`exitAtFirstDate()` resets cohort end date based on a set of specified column dates. The first date that occurs is chosen.

Usage

```
exitAtFirstDate(
  cohort,
  dateColumns,
  cohortId = NULL,
  returnReason = TRUE,
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`dateColumns`	Date columns in the cohort table to consider.
`cohortId`	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
`returnReason`	If TRUE it will return a column stating which column in `dateColumns` is used as a new cohort end date.
`name`	Name of the new cohort with the restriction.

Value

The cohort table.

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor(tables = list(
"cohort" = dplyr::tibble(
  cohort_definition_id = 1,
  subject_id = c(1, 2, 3, 4),
  cohort_start_date = as.Date(c("2000-06-03", "2000-01-01", "2015-01-15", "2000-12-09")),
```

```
  cohort_end_date = as.Date(c("2001-09-01", "2001-01-12", "2015-02-15", "2002-12-09")),
  date_1 = as.Date(c("2001-08-01", "2001-01-01", "2015-01-15", "2002-12-09")),
  date_2 = as.Date(c("2001-08-01", NA, "2015-04-15", "2002-12-09"))
)
))
cdm$cohort |> exitAtFirstDate(dateColumns = c("date_1", "date_2"))
```

`exitAtLastDate` *Set cohort end date to the last of a set of column dates*

Description

`exitAtLastDate()` resets cohort end date based on a set of specified column dates. The last date that occurs is chosen.

Usage

```
exitAtLastDate(
  cohort,
  dateColumns,
  cohortId = NULL,
  returnReason = TRUE,
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`dateColumns`	description
`cohortId`	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
`returnReason`	If TRUE it will return a column stating which column in `dateColumns` is used as a new cohort end date. description
`name`	Name of the new cohort with the restriction.

Value

The cohort table.

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor(tables = list(
"cohort" = dplyr::tibble(
  cohort_definition_id = 1,
  subject_id = c(1, 2, 3, 4),
```

```
  cohort_start_date = as.Date(c("2000-06-03", "2000-01-01", "2015-01-15", "2000-12-09")),
  cohort_end_date = as.Date(c("2001-09-01", "2001-01-12", "2015-02-15", "2002-12-09")),
  date_1 = as.Date(c("2001-08-01", "2001-01-01", "2015-01-15", "2002-12-09")),
  date_2 = as.Date(c("2001-08-01", NA, "2015-04-15", "2002-12-09"))
)
))
cdm$cohort |> exitAtLastDate(dateColumns = c("date_1", "date_2"))
```

`exitAtObservationEnd` *Set cohort end date to end of observation*

Description

`exitAtObservationEnd()` resets cohort end date based on a set of specified column dates. The last date that occurs is chosen.

This functions changes cohort end date to the end date of the observation period corresponding to the cohort entry. In the case were this generates overlapping records in the cohort, overlapping entries will be merged.

Usage

```
exitAtObservationEnd(cohort, cohortId = NULL, name = tableName(cohort))
```

Arguments

`cohort`	A cohort table in a cdm reference.
`cohortId`	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
`name`	Name of the new cohort with the restriction.

Value

The cohort table.

Examples

```
library(CohortConstructor)

cdm <- mockCohortConstructor()
cdm$cohort1 |> exitAtObservationEnd()
```

`intersectCohorts` *Generate a combination cohort set between the intersection of different cohorts.*

Description

`intersectCohorts()` combines different cohort entries, with those records that overlap combined and kept. Cohort entries are when an individual was in *both* of the cohorts.

Usage

```
intersectCohorts(
  cohort,
  cohortId = NULL,
  gap = 0,
  mutuallyExclusive = FALSE,
  returnOnlyComb = FALSE,
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`cohortId`	IDs of the cohorts to include. If NULL all cohorts will be considered. Cohorts not included will be removed from the cohort set.
`gap`	Number of days between two subsequent cohort entries to be merged in a single cohort record.
`mutuallyExclusive`	Whether the generated cohorts are mutually exclusive or not.
`returnOnlyComb`	Whether to only get the combination cohort back
`name`	Name of the new cohort with the demographic requirements.

Value

A cohort table.

Examples

```
library(CohortConstructor)

cdm <- mockCohortConstructor(nPerson = 100)

cdm$cohort3 <- intersectCohorts(
  cohort = cdm$cohort2,
  name = "cohort3",
)
```

```
settings(cdm$cohort3)
```

`matchCohorts` *Generate a new cohort matched cohort*

Description

`matchCohorts()` generate a new cohort matched to individuals in an existing cohort. Individuals can be matched based on year of birth and sex.

Usage

```
matchCohorts(
  cohort,
  cohortId = NULL,
  matchSex = TRUE,
  matchYearOfBirth = TRUE,
  ratio = 1,
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`cohortId`	IDs of the cohorts to include. If NULL all cohorts will be considered. Cohorts not included will be removed from the cohort set.
`matchSex`	Whether to match in sex.
`matchYearOfBirth`	Whether to match in year of birth.
`ratio`	Number of allowed matches per individual in the target cohort.
`name`	Name of the new generated cohort set.

Value

A cohort table.

Examples

```
library(CohortConstructor)
library(dplyr)
cdm <- mockCohortConstructor(nPerson = 200)
cdm$new_matched_cohort <- cdm$cohort2 |>
  matchCohorts(
    name = "new_matched_cohort",
    cohortId = 2,
```

```
    matchSex = TRUE,
    matchYearOfBirth = TRUE,
    ratio = 1)
cdm$new_matched_cohort
```

`measurementCohort` *Create cohorts measurement based cohorts*

Description

`measurementCohort()` creates cohorts based on patient records contained in the measurement table. This function extends the `conceptCohort()` as it allows for measurement values associated with the records to be specified.

- If `valueAsConcept` and `valueAsNumber` are NULL then no requirements on of the values associated with measurement records and using `measurementCohort()` will lead to the same result as using `conceptCohort()` (so long as all concepts are from the measurement domain).
- If one of `valueAsConcept` and `valueAsNumber` is not NULL then records will be required to have values that satisfy the requirement specified.
- If both `valueAsConcept` and `valueAsNumber` are not NULL, records will be required to have values that fulfill *either* of the requirements

Usage

```
measurementCohort(
  cdm,
  conceptSet,
  name,
  valueAsConcept = NULL,
  valueAsNumber = NULL
)
```

Arguments

`cdm`	A cdm reference.
`conceptSet`	A conceptSet, which can either be a codelist or a conceptSetExpression.
`name`	Name of the cohort in the cdm object.
`valueAsConcept`	A vector of cohort IDs used to filter measurements. Only measurements with these values in the `value_as_concept_id` column of the measurement table will be included. If NULL all entries independently of their value as concept will be considered.
`valueAsNumber`	A named list indicating the range of values and the unit they correspond to, as follows: list("unit_concept_id" = c(rangeValue1, rangeValue2)). If NULL, all entries independently of their value as number will be included.

Value

A cohort table

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor(con = NULL)
cdm$concept <- cdm$concept |>
  dplyr::union_all(
    dplyr::tibble(
      concept_id = c(4326744, 4298393, 45770407, 8876, 4124457),
      concept_name = c("Blood pressure", "Systemic blood pressure",
                       "Baseline blood pressure", "millimeter mercury column",
                       "Normal range"),
      domain_id = "Measurement",
      vocabulary_id = c("SNOMED", "SNOMED", "SNOMED", "UCUM", "SNOMED"),
      standard_concept = "S",
      concept_class_id = c("Observable Entity", "Observable Entity",
                           "Observable Entity", "Unit", "Qualifier Value"),
      concept_code = NA,
      valid_start_date = NA,
      valid_end_date = NA,
      invalid_reason = NA
    )
  )
cdm$measurement <- dplyr::tibble(
  measurement_id = 1:4,
  person_id = c(1, 1, 2, 3),
  measurement_concept_id = c(4326744, 4298393, 4298393, 45770407),
  measurement_date = as.Date(c("2000-07-01", "2000-12-11", "2002-09-08",
                               "2015-02-19")),
  measurement_type_concept_id = NA,
  value_as_number = c(100, 125, NA, NA),
  value_as_concept_id = c(0, 0, 0, 4124457),
  unit_concept_id = c(8876, 8876, 0, 0)
)
cdm <- CDMConnector::copyCdmTo(
  con = DBI::dbConnect(duckdb::duckdb()),
  cdm = cdm, schema = "main")

cdm$cohort <- measurementCohort(
  cdm = cdm,
  name = "cohort",
  conceptSet = list("normal_blood_pressure" = c(4326744, 4298393, 45770407)),
  valueAsConcept = c(4124457),
  valueAsNumber = list("8876" = c(70, 120))
)

cdm$cohort
```

`mockCohortConstructor` *Function to create a mock cdm reference for CohortConstructor*

Description

`mockCohortConstructor()` creates an example dataset that can be used for demonstrating and testing the package

Usage

```
mockCohortConstructor(
  nPerson = 10,
  conceptTable = NULL,
  tables = NULL,
  conceptId = NULL,
  conceptIdClass = NULL,
  drugExposure = FALSE,
  conditionOccurrence = FALSE,
  measurement = FALSE,
  death = FALSE,
  otherTables = NULL,
  con = DBI::dbConnect(duckdb::duckdb()),
  writeSchema = "main",
  seed = 123
)
```

Arguments

`nPerson`	number of person in the cdm
`conceptTable`	user defined concept table
`tables`	list of tables to include in the cdm
`conceptId`	list of concept id
`conceptIdClass`	the domain class of the conceptId
`drugExposure`	T/F include drug exposure table in the cdm
`conditionOccurrence`	T/F include condition occurrence in the cdm
`measurement`	T/F include measurement in the cdm
`death`	T/F include death table in the cdm
`otherTables`	it takes a list of single tibble with names to include other tables in the cdm
`con`	A DBI connection to create the cdm mock object.
`writeSchema`	Name of an schema on the same connection with writing permissions.
`seed`	Seed passed to omock::mockCdmFromTable

Value

cdm object

Examples

```
library(CohortConstructor)

cdm <- mockCohortConstructor()

cdm
```

`requireAge` *Restrict cohort on age*

Description

`requireAge()` filters cohort records, keeping only records where individuals satisfy the specified age criteria.

Usage

```
requireAge(
  cohort,
  ageRange,
  cohortId = NULL,
  indexDate = "cohort_start_date",
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`ageRange`	A list of minimum and maximum age.
`cohortId`	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
`indexDate`	Variable in cohort that contains the date to compute the demographics characteristics on which to restrict on.
`name`	Name of the new cohort with the age requirement.

Value

The cohort table with only records for individuals satisfying the age requirement

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor()
cdm$cohort1 |>
  requireAge(indexDate = "cohort_start_date",
             ageRange = list(c(18, 65)))
```

`requireCohortIntersect` *Require cohort subjects are present (or absence) in another cohort*

Description

`requireCohortIntersect()` filters a cohort table based on a requirement that an individual is seen (or not seen) in another cohort in some time window around an index date.

Usage

```
requireCohortIntersect(
  cohort,
  targetCohortTable,
  window,
  intersections = c(1, Inf),
  cohortId = NULL,
  targetCohortId = NULL,
  indexDate = "cohort_start_date",
  targetStartDate = "cohort_start_date",
  targetEndDate = "cohort_end_date",
  censorDate = NULL,
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`targetCohortTable`	Name of the cohort that we want to check for intersect.
`window`	Window to consider events over.
`intersections`	A range indicating number of intersections for criteria to be fulfilled. If a single number is passed, the number of intersections must match this.
`cohortId`	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
`targetCohortId`	Vector of cohort definition ids to include.
`indexDate`	Variable in x that contains the date to compute the intersection.

targetStartDate
: Date of reference in cohort table, either for start (in overlap) or on its own (for incidence).

targetEndDate
: Date of reference in cohort table, either for end (overlap) or NULL (if incidence).

censorDate
: Whether to censor overlap events at a specific date or a column date of x.

name
: Name of the new cohort with the future observation restriction.

Value

Cohort table with only those isatisfying the criteria kept

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor()
cdm$cohort1 |>
  requireCohortIntersect(targetCohortTable = "cohort2",
                         targetCohortId = 1,
                         indexDate = "cohort_start_date",
                         window = c(-Inf, 0))
```

---

requireConceptIntersect

Require cohort subjects to have (or not have) events of a concept list

---

Description

`requireConceptIntersect()` filters a cohort table based on a requirement that an individual is seen (or not seen) to have events related to a concept list in some time window around an index date.

Usage

```
requireConceptIntersect(
  cohort,
  conceptSet,
  window,
  intersections = c(1, Inf),
  cohortId = NULL,
  indexDate = "cohort_start_date",
  targetStartDate = "event_start_date",
  targetEndDate = "event_end_date",
  censorDate = NULL,
  name = tableName(cohort)
)
```

Arguments

cohort	A cohort table in a cdm reference.
conceptSet	Concept set list.
window	Window to consider events over.
intersections	A range indicating number of intersections for criteria to be fulfilled. If a single number is passed, the number of intersections must match this.
cohortId	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged..
indexDate	Variable in x that contains the date to compute the intersection.
targetStartDate	Date of reference in cohort table, either for start (in overlap) or on its own (for incidence).
targetEndDate	Date of reference in cohort table, either for end (overlap) or NULL (if incidence).
censorDate	Whether to censor overlap events at a specific date or a column date of x.
name	Name of the new cohort with the future observation restriction.

Value

Cohort table with only those with the events in the concept list kept (or those without the event if negate = TRUE)

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor(conditionOccurrence = TRUE)
cdm$cohort2 <-  requireConceptIntersect(
  cohort = cdm$cohort1,
  conceptSet = list(a = 1),
  window = c(-Inf, 0),
  name = "cohort2")
```

---

requireDeathFlag *Require cohort subjects have (or do not have) a death record*

Description

`requireDeathFlag()` filters a cohort table based on a requirement that an individual is seen (or not seen) to have a death in some time window around an index date.

Usage

```
requireDeathFlag(
  cohort,
  window,
  cohortId = NULL,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  negate = FALSE,
  name = tableName(cohort)
)
```

Arguments

cohort	A cohort table in a cdm reference.
window	Window to consider events over.
cohortId	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
indexDate	Variable in x that contains the date to compute the intersection.
censorDate	Whether to censor overlap events at a specific date or a column date of x.
negate	If set as TRUE, criteria will be applied as exclusion rather than inclusion (i.e. require absence in another cohort).
name	Name of the new cohort with the future observation restriction.

Value

Cohort table with only those with a death event kept (or without if negate = TRUE)

Examples

```
library(CDMConnector)
library(CohortConstructor)
cdm <- mockCohortConstructor(death = TRUE)
cdm$cohort1 <- cdm$cohort1 |> requireDeathFlag(window = list(c(0, Inf)))
attrition(cdm$cohort1)
```

---

requireDemographics *Restrict cohort on patient demographics*

Description

`requireDemographics()` filters cohort records, keeping only records where individuals satisfy the specified demographic criteria.

Usage

```
requireDemographics(
  cohort,
  cohortId = NULL,
  indexDate = "cohort_start_date",
  ageRange = list(c(0, 150)),
  sex = c("Both"),
  minPriorObservation = 0,
  minFutureObservation = 0,
  requirementInteractions = TRUE,
  name = tableName(cohort)
)
```

Arguments

cohort	A cohort table in a cdm reference.
cohortId	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
indexDate	Variable in cohort that contains the date to compute the demographics characteristics on which to restrict on.
ageRange	A list of minimum and maximum age.
sex	Can be "Both", "Male" or "Female". If one of the latter, only those with that sex will be included.
minPriorObservation	A minimum number of prior observation days in the database.
minFutureObservation	A minimum number of future observation days in the database.
requirementInteractions	If TRUE, cohorts will be created for all combinations of ageGroup, sex, and daysPriorObservation. If FALSE, only the first value specified for the other factors will be used. Consequently, order of values matters when requirementInteractions is FALSE.
name	Name of the new cohort with the demographic requirements.

Value

The cohort table with only records for individuals satisfying the demographic requirements

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor(nPerson = 100)
cdm$cohort1 |>
  requireDemographics(indexDate = "cohort_start_date",
                      ageRange = list(c(18, 65)),
                      sex = "Female",
                      minPriorObservation = 365)
```

`requireFutureObservation` *Restrict cohort on future observation*

Description

`requireFutureObservation()` filters cohort records, keeping only records where individuals satisfy the specified future observation criteria.

Usage

```
requireFutureObservation(
  cohort,
  minFutureObservation,
  cohortId = NULL,
  indexDate = "cohort_start_date",
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`minFutureObservation`	A minimum number of future observation days in the database.
`cohortId`	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
`indexDate`	Variable in cohort that contains the date to compute the demographics characteristics on which to restrict on.
`name`	Name of the new cohort with the future observation restriction.

Value

The cohort table with only records for individuals satisfying the future observation requirement

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor()
cdm$cohort1 |>
  requireFutureObservation(indexDate = "cohort_start_date",
                           minFutureObservation = 30)
```

`requireInDateRange` *Require that an index date is within a date range*

Description

`requireInDateRange()` filters cohort records, keeping only those for which the index date is within the specified date range.

Usage

```
requireInDateRange(
  cohort,
  dateRange,
  cohortId = NULL,
  indexDate = "cohort_start_date",
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`dateRange`	A window of time during which the index date must have been observed.
`cohortId`	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
`indexDate`	Variable in cohort that contains the index date of interest
`name`	Name of the new cohort with the restriction.

Value

The cohort table with any cohort entries outside of the date range dropped

Examples

```
library(CohortConstructor)

cdm <- mockCohortConstructor(nPerson = 100)
cdm$cohort1 |>
  requireInDateRange(indexDate = "cohort_start_date",
                     dateRange = as.Date(c("2010-01-01", "2019-01-01")))
```

requireIsFirstEntry *Restrict cohort to first entry*

Description

`requireIsFirstEntry()` filters cohort records, keeping only the first cohort entry per person.

Usage

```
requireIsFirstEntry(
  cohort,
  cohortId = NULL,
  indexDate = "cohort_start_date",
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`cohortId`	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
`indexDate`	Column name in cohort that contains the date to restrict on.
`name`	Name of the new cohort with the restriction.

Value

A cohort table in a cdm reference.

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor()
cdm$cohort1 <- requireIsFirstEntry(cdm$cohort1)
```

requireIsLastEntry *Restrict cohort to last entry per person*

Description

`requireIsLastEntry()` filters cohort records, keeping only the last cohort entry per person.

Usage

```
requireIsLastEntry(
  cohort,
  cohortId = NULL,
  indexDate = "cohort_start_date",
  name = tableName(cohort)
)
```

Arguments

cohort	A cohort table in a cdm reference.
cohortId	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
indexDate	Column name in cohort that contains the date to restrict on.
name	Name of the new cohort with the restriction.

Value

A cohort table in a cdm reference.

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor()
cdm$cohort1 <- requireIsLastEntry(cdm$cohort1)
```

---

requirePriorObservation

Restrict cohort on prior observation

Description

`requirePriorObservation()` filters cohort records, keeping only records where individuals satisfy the specified prior observation criteria.

Usage

```
requirePriorObservation(
  cohort,
  minPriorObservation,
  cohortId = NULL,
  indexDate = "cohort_start_date",
  name = tableName(cohort)
)
```

Arguments

cohort	A cohort table in a cdm reference.
minPriorObservation	A minimum number of prior observation days in the database.
cohortId	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
indexDate	Variable in cohort that contains the date to compute the demographics characteristics on which to restrict on.
name	Name of the new cohort with the prior observation restriction.

Value

The cohort table with only records for individuals satisfying the prior observation requirement

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor()
cdm$cohort1 |>
  requirePriorObservation(indexDate = "cohort_start_date",
                          minPriorObservation = 365)
```

---

requireSex *Restrict cohort on sex*

Description

`requireSex()` filters cohort records, keeping only records where individuals satisfy the specified sex criteria.

Usage

```
requireSex(cohort, sex, cohortId = NULL, name = tableName(cohort))
```

Arguments

cohort	A cohort table in a cdm reference.
sex	Can be "Both", "Male" or "Female". If one of the latter, only those with that sex will be included.
cohortId	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
name	Name of the new cohort with the sex requirements.

Value

The cohort table with only records for individuals satisfying the sex requirement

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor()
cdm$cohort1 |>
  requireSex(sex = "Female")
```

`requireTableIntersect` *Require cohort subjects are present in another clinical table*

Description

`requireTableIntersect()` filters a cohort table based on a requirement that an individual is seen (or not seen) to have a record (or no records) in a clinical table in some time window around an index date.

Usage

```
requireTableIntersect(
  cohort,
  tableName,
  window,
  intersections = c(1, Inf),
  cohortId = NULL,
  indexDate = "cohort_start_date",
  targetStartDate = startDateColumn(tableName),
  targetEndDate = endDateColumn(tableName),
  censorDate = NULL,
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`tableName`	Name of the table to check for intersect.
`window`	Window to consider events over.
`intersections`	A range indicating number of intersections for criteria to be fulfilled. If a single number is passed, the number of intersections must match this.
`cohortId`	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
`indexDate`	Variable in x that contains the date to compute the intersection.
`targetStartDate`	Date of reference in cohort table, either for start (in overlap) or on its own (for incidence).

targetEndDate	Date of reference in cohort table, either for end (overlap) or NULL (if incidence).
censorDate	Whether to censor overlap events at a specific date or a column date of x.
name	Name of the new cohort with the future observation restriction.

Value

Cohort table with only those in the other table kept (or those that are not in the table if negate = TRUE)

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor(drugExposure = TRUE)
cdm$cohort1 |>
  requireTableIntersect(tableName = "drug_exposure",
                            indexDate = "cohort_start_date",
                            window = c(-Inf, 0))
```

---

sampleCohorts *Sample a cohort table for a given number of individuals.*

Description

sampleCohorts() samples an existing cohort table for a given number of people. All records of these individuals are preserved.

Usage

```
sampleCohorts(cohort, cohortId = NULL, n, name = tableName(cohort))
```

Arguments

cohort	A cohort table in a cdm reference.
cohortId	IDs of the cohorts to include. If NULL all cohorts will be considered. Cohorts not included will not be sampled.
n	Number of people to be sampled for each included cohort.
name	Name of the new sampled cohort.

Value

Cohort table with the specified cohorts sampled.

Examples

```
library(CohortConstructor)

cdm <- mockCohortConstructor(nPerson = 100)

cdm$cohort2 |> sampleCohorts(cohortId = 1, n = 10)
```

`stratifyCohorts` *Create a new cohort table from stratifying an existing one*

Description

`stratifyCohorts()` creates new cohorts, splitting an existing cohort based on specified columns on which to stratify on.

Usage

```
stratifyCohorts(
  cohort,
  strata,
  cohortId = NULL,
  removeStrata = TRUE,
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`strata`	A strata list that point to columns in cohort table.
`cohortId`	IDs of the cohorts to include. If NULL all cohorts will be considered. Cohorts not included will be removed from the cohort set.
`removeStrata`	Whether to remove strata columns from final cohort table.
`name`	Name of the new cohort.

Value

Cohort table stratified.

Examples

```
library(CohortConstructor)
library(PatientProfiles)

cdm <- mockCohortConstructor()

cdm$my_cohort <- cdm$cohort1 |>
```

```
  addAge(ageGroup = list("child" = c(0, 17), "adult" = c(18, Inf))) |>
  addSex() |>
  stratifyCohorts(
    strata = list("sex", c("sex", "age_group")), name = "my_cohort"
  )

cdm$my_cohort

settings(cdm$my_cohort)

attrition(cdm$my_cohort)
```

`subsetCohorts` *Generate a cohort table using a subset of cohorts from another table.*

Description

`subsetCohorts()` filters an existing cohort table, keeping only the records from cohorts that are specified.

Usage

```
subsetCohorts(cohort, cohortId, minCohortCount = 0, name = tableName(cohort))
```

Arguments

`cohort`	A cohort table in a cdm reference.
`cohortId`	IDs of the cohorts to include. If NULL all cohorts will be considered. Cohorts not included will be removed from the cohort set.
`minCohortCount`	the minimum count of a cohort to be included. Default is 0, meaning all non-empty cohorts will be included. Cohorts not included will be removed from the cohort set.
`name`	Name of the new cohort with the demographic requirements.

Value

Cohort table with only cohorts in cohortId.

Examples

```
library(CohortConstructor)

cdm <- mockCohortConstructor(nPerson = 100)

cdm$cohort1 |> subsetCohorts(cohortId = 1)
```

`trimDemographics` *Restrict cohort on patient demographics*

Description

`trimDemographics()` resets the cohort start and end date based on the specified demographic criteria is satisfied.

Usage

```
trimDemographics(
  cohort,
  cohortId = NULL,
  ageRange = NULL,
  sex = NULL,
  minPriorObservation = NULL,
  minFutureObservation = NULL,
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`cohortId`	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
`ageRange`	A list of minimum and maximum age.
`sex`	Can be "Both", "Male" or "Female". If one of the latter, only those with that sex will be included.
`minPriorObservation`	A minimum number of prior observation days in the database.
`minFutureObservation`	A minimum number of future observation days in the database.
`name`	Name of the new cohort with the demographic requirements.

Value

The cohort table with only records for individuals satisfying the demographic requirements

Examples

```
library(CohortConstructor)

cdm <- mockCohortConstructor(nPerson = 100)

cdm$cohort1 |> trimDemographics(ageRange = list(c(10, 30)))
```

`trimToDateRange` *Trim cohort dates to be within a date range*

Description

`trimToDateRange()` resets the cohort start and end date based on the specified date range.

Usage

```
trimToDateRange(
  cohort,
  dateRange,
  cohortId = NULL,
  startDate = "cohort_start_date",
  endDate = "cohort_end_date",
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`dateRange`	A window of time during which the index date must have been observed.
`cohortId`	IDs of the cohorts to modify. If NULL, all cohorts will be used; otherwise, only the specified cohorts will be modified, and the rest will remain unchanged.
`startDate`	Variable with earliest date.
`endDate`	Variable with latest date.
`name`	Name of the new cohort with the restriction.

Value

The cohort table with record timings updated to only be within the date range. Any records with all time outside of the range will have been dropped.

Examples

```
library(CohortConstructor)
cdm <- mockCohortConstructor()
cdm$cohort1 |>
  trimToDateRange(startDate = "cohort_start_date",
                  endDate = "cohort_end_date",
                  dateRange = as.Date(c("2015-01-01",
                                        "2015-12-31")))
```

`unionCohorts` *Generate cohort from the union of different cohorts*

Description

`unionCohorts()` combines different cohort entries, with those records that overlap combined and kept. Cohort entries are when an individual was in *either* of the cohorts.

Usage

```
unionCohorts(
  cohort,
  cohortId = NULL,
  gap = 0,
  cohortName = NULL,
  name = tableName(cohort)
)
```

Arguments

`cohort`	A cohort table in a cdm reference.
`cohortId`	IDs of the cohorts to include. If NULL all cohorts will be considered. Cohorts not included will be removed from the cohort set.
`gap`	Number of days between two subsequent cohort entries of a subject that will be merged in a single cohort entry
`cohortName`	Name of the returned cohort. If NULL, the cohort name will be created by collapsing the individual cohort names, separated by "_".
`name`	Name of the new cohort table.

Value

A cohort table.

Examples

```
library(CohortConstructor)

cdm <- mockCohortConstructor(nPerson = 100)

cdm$cohort2 <- cdm$cohort2 |> unionCohorts()
settings(cdm$cohort2)
```

yearCohorts *Generate a new cohort table restricting cohort entries to certain years*

Description

`yearCohorts()` splits a cohort into multiple cohorts, one for each year.

Usage

```
yearCohorts(cohort, years, cohortId = NULL, name = tableName(cohort))
```

Arguments

cohort	A cohort table in a cdm reference.
years	Numeric vector of years to use to restrict observation to.
cohortId	IDs of the cohorts to include. If NULL all cohorts will be considered. Cohorts not included will be removed from the cohort set.
name	Name of the new cohort table.

Value

A cohort table.

Examples

```
library(CohortConstructor)

cdm <- mockCohortConstructor(nPerson = 100)

cdm$cohort1 <- cdm$cohort1 |> yearCohorts(years = 2000:2002)
settings(cdm$cohort1)
```

Index