Package: PatientProfiles (via r-universe)

October 26, 2024

Type Package

Title Identify Characteristics of Patients in the OMOP Common Data Model

Version 1.2.1

Maintainer Marti Catala <marti.catalasabate@ndorms.ox.ac.uk>

Description Identify the characteristics of patients in data mapped to the Observational Medical Outcomes Partnership (OMOP) common data model.

License Apache License (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

Suggests covr, duckdb (>= 0.9.0), testthat (>= 3.1.5), knitr, CodelistGenerator, rmarkdown, glue, odbc, ggplot2, spelling, RPostgres, dbplyr, here, DT, DBI, gt, tictoc, withr, scales

Imports magrittr, CDMConnector (>= 1.3.1), dplyr, tidyr, rlang, cli, stringr, omopgenerics (>= 0.2.0), visOmopResults (>= 0.2.0), purrr, lifecycle

URL https://darwin-eu-dev.github.io/PatientProfiles/

BugReports https://github.com/darwin-eu-dev/PatientProfiles/issues

Language en-US

Depends R (>= 2.10)

Config/testthat/edition 3

Config/testthat/parallel true

VignetteBuilder knitr

NeedsCompilation no

Author Marti Catala [aut, cre] (<https://orcid.org/0000-0003-3308-9905>), Yuchen Guo [aut] (<https://orcid.org/0000-0002-0847-4855>), Mike Du [aut] (<https://orcid.org/0000-0002-9517-8834>), Kim Lopez-Guell [aut] (<https://orcid.org/0000-0002-8462-8668>), Edward Burn

[aut] (<https://orcid.org/0000-0002-9286-1128>), Nuria
Mercade-Besora [aut] (<https://orcid.org/0009-0006-7948-3747>),
Xintong Li [ctb] (<https://orcid.org/0000-0002-6872-5804>),
Xihang Chen [ctb] (<https://orcid.org/0009-0001-8112-8959>)

Repository CRAN

Date/Publication 2024-10-25 04:40:02 UTC

Contents

---

addAge *Compute the age of the individuals at a certain date*

Description

Compute the age of the individuals at a certain date

Usage

```
addAge(
  x,
  indexDate = "cohort_start_date",
  ageName = "age",
  ageGroup = NULL,
  ageMissingMonth = 1,
  ageMissingDay = 1,
  ageImposeMonth = FALSE,
  ageImposeDay = FALSE,
  missingAgeGroupValue = "None",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the date to compute the age.
ageName	Name of the new column that contains age.
ageGroup	List of age groups to be added.
ageMissingMonth	Month of the year assigned to individuals with missing month of birth. By default: 1.
ageMissingDay	day of the month assigned to individuals with missing day of birth. By default: 1.
ageImposeMonth	Whether the month of the date of birth will be considered as missing for all the individuals.
ageImposeDay	Whether the day of the date of birth will be considered as missing for all the individuals.

missingAgeGroupValue
: Value to include if missing age.

name
: Name of the new table, if NULL a temporary table is returned.

Value

tibble with the age column added.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 |>
  addAge()
mockDisconnect(cdm = cdm)
```

---

addAgeQuery *Query to add the age of the individuals at a certain date*

Description

‘r lifecycle::badge("experimental")‘ Same as ‘addAge()‘, except query is not computed to a table.

Usage

```
addAgeQuery(
  x,
  indexDate = "cohort_start_date",
  ageName = "age",
  ageGroup = NULL,
  ageMissingMonth = 1,
  ageMissingDay = 1,
  ageImposeMonth = FALSE,
  ageImposeDay = FALSE,
  missingAgeGroupValue = "None"
)
```

Arguments

x
: Table with individuals in the cdm.

indexDate
: Variable in x that contains the date to compute the age.

ageName
: Name of the new column that contains age.

ageGroup
: List of age groups to be added.

ageMissingMonth
: Month of the year assigned to individuals with missing month of birth. By default: 1.

ageMissingDay	day of the month assigned to individuals with missing day of birth. By default: 1.
ageImposeMonth	Whether the month of the date of birth will be considered as missing for all the individuals.
ageImposeDay	Whether the day of the date of birth will be considered as missing for all the individuals.
missingAgeGroupValue	Value to include if missing age.

Value

tibble with the age column added.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 |>
  addAgeQuery()
mockDisconnect(cdm = cdm)
```

---

addCategories *Categorize a numeric variable*

Description

Categorize a numeric variable

Usage

```
addCategories(
  x,
  variable,
  categories,
  missingCategoryValue = "None",
  overlap = FALSE,
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
variable	Target variable that we want to categorize.
categories	List of lists of named categories with lower and upper limit.

missingCategoryValue
: Value to assign to those individuals not in any named category. If NULL or NA, missing will values will be given.

overlap
: TRUE if the categories given overlap.

name
: Name of the new table, if NULL a temporary table is returned.

Value

The x table with the categorical variable added.

Examples

```
cdm <- mockPatientProfiles()

result <- cdm$cohort1 %>%
  addAge() %>%
  addCategories(
    variable = "age",
    categories = list("age_group" = list(
      "0 to 39" = c(0, 39), "40 to 79" = c(40, 79), "80 to 150" = c(80, 150)
    ))
  )
mockDisconnect(cdm = cdm)
```

---

addCdmName *Add cdm name*

Description

Add cdm name

Usage

```
addCdmName(table, cdm = omopgenerics::cdmReference(table))
```

Arguments

table
: Table in the cdm

cdm
: A cdm reference object

Value

Table with an extra column with the cdm names

Examples

```
library(PatientProfiles)

cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addCdmName()
```

`addCohortIntersectCount` *It creates columns to indicate number of occurrences of intersection with a cohort*

Description

It creates columns to indicate number of occurrences of intersection with a cohort

Usage

```
addCohortIntersectCount(
  x,
  targetCohortTable,
  targetCohortId = NULL,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  targetStartDate = "cohort_start_date",
  targetEndDate = "cohort_end_date",
  window = list(c(0, Inf)),
  nameStyle = "{cohort_name}_{window_name}",
  name = NULL
)
```

Arguments

`x`	Table with individuals in the cdm.
`targetCohortTable`	name of the cohort that we want to check for overlap.
`targetCohortId`	vector of cohort definition ids to include.
`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.
`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).
`window`	window to consider events of.

nameStyle	naming of the added column or columns, should include required parameters.
name	Name of the new table, if NULL a temporary table is returned.

Value

table with added columns with overlap information.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addCohortIntersectCount(
    targetCohortTable = "cohort2"
  )
mockDisconnect(cdm = cdm)
```

---

addCohortIntersectDate

Date of cohorts that are present in a certain window

Description

Date of cohorts that are present in a certain window

Usage

```
addCohortIntersectDate(
  x,
  targetCohortTable,
  targetCohortId = NULL,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  targetDate = "cohort_start_date",
  order = "first",
  window = c(0, Inf),
  nameStyle = "{cohort_name}_{window_name}",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
targetCohortTable	Cohort table to.

targetCohortId	Cohort IDs of interest from the other cohort table. If NULL, all cohorts will be used with a time variable added for each cohort of interest.
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.
targetDate	Date of interest in the other cohort table. Either cohort_start_date or cohort_end_date.
order	date to use if there are multiple records for an individual during the window of interest. Either first or last.
window	Window of time to identify records relative to the indexDate. Records outside of this time period will be ignored.
nameStyle	naming of the added column or columns, should include required parameters.
name	Name of the new table, if NULL a temporary table is returned.

Value

x along with additional columns for each cohort of interest.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addCohortIntersectDate(
    targetCohortTable = "cohort2"
  )
mockDisconnect(cdm = cdm)
```

---

addCohortIntersectDays

It creates columns to indicate the number of days between the current table and a target cohort

---

Description

It creates columns to indicate the number of days between the current table and a target cohort

Usage

```
addCohortIntersectDays(
  x,
  targetCohortTable,
  targetCohortId = NULL,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  targetDate = "cohort_start_date",
```

```
  order = "first",
  window = c(0, Inf),
  nameStyle = "{cohort_name}_{window_name}",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
targetCohortTable	Cohort table to.
targetCohortId	Cohort IDs of interest from the other cohort table. If NULL, all cohorts will be used with a days variable added for each cohort of interest.
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.
targetDate	Date of interest in the other cohort table. Either cohort_start_date or cohort_end_date.
order	date to use if there are multiple records for an individual during the window of interest. Either first or last.
window	Window of time to identify records relative to the indexDate. Records outside of this time period will be ignored.
nameStyle	naming of the added column or columns, should include required parameters.
name	Name of the new table, if NULL a temporary table is returned.

Value

x along with additional columns for each cohort of interest.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addCohortIntersectDays(
    targetCohortTable = "cohort2"
  )
mockDisconnect(cdm = cdm)
```

addCohortIntersectFlag *It creates columns to indicate the presence of cohorts*

Description

It creates columns to indicate the presence of cohorts

Usage

```
addCohortIntersectFlag(
  x,
  targetCohortTable,
  targetCohortId = NULL,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  targetStartDate = "cohort_start_date",
  targetEndDate = "cohort_end_date",
  window = list(c(0, Inf)),
  nameStyle = "{cohort_name}_{window_name}",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
targetCohortTable	name of the cohort that we want to check for overlap.
targetCohortId	vector of cohort definition ids to include.
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.
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).
window	window to consider events of.
nameStyle	naming of the added column or columns, should include required parameters.
name	Name of the new table, if NULL a temporary table is returned.

Value

table with added columns with overlap information.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addCohortIntersectFlag(
    targetCohortTable = "cohort2"
  )
mockDisconnect(cdm = cdm)
```

---

addCohortName *Add cohort name for each cohort_definition_id*

Description

Add cohort name for each cohort_definition_id

Usage

```
addCohortName(cohort)
```

Arguments

cohort	cohort to which add the cohort name

Value

cohort with an extra column with the cohort names

Examples

```
library(PatientProfiles)

cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addCohortName()
```

addConceptIntersectCount *It creates column to indicate the count overlap information between a table and a concept*

Description

It creates column to indicate the count overlap information between a table and a concept

Usage

```
addConceptIntersectCount(
  x,
  conceptSet,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetStartDate = "event_start_date",
  targetEndDate = "event_end_date",
  inObservation = TRUE,
  nameStyle = "{concept_name}_{window_name}",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
conceptSet	Concept set list.
indexDate	Variable in x that contains the date to compute the intersection.
censorDate	whether to censor overlap events at a date column of x
window	window to consider events in.
targetStartDate	Event start date to use for the intersection.
targetEndDate	Event end date to use for the intersection.
inObservation	If TRUE only records inside an observation period will be considered.
nameStyle	naming of the added column or columns, should include required parameters.
name	Name of the new table, if NULL a temporary table is returned.

Value

table with added columns with overlap information

Examples

```
library(PatientProfiles)
cdm <- mockPatientProfiles()
concept <- dplyr::tibble(
  concept_id = c(1125315),
  domain_id = "Drug",
  vocabulary_id = NA_character_,
  concept_class_id = "Ingredient",
  standard_concept = "S",
  concept_code = NA_character_,
  valid_start_date = as.Date("1900-01-01"),
  valid_end_date = as.Date("2099-01-01"),
  invalid_reason = NA_character_
) %>%
  dplyr::mutate(concept_name = paste0("concept: ", .data$concept_id))
cdm <- CDMConnector::insertTable(cdm, "concept", concept)
result <- cdm$cohort1 %>%
  addConceptIntersectCount(
    conceptSet = list("acetaminophen" = 1125315)
  ) %>%
  dplyr::collect()
mockDisconnect(cdm = cdm)
```

---

addConceptIntersectDate

It creates column to indicate the date overlap information between a table and a concept

---

Description

It creates column to indicate the date overlap information between a table and a concept

Usage

```
addConceptIntersectDate(
  x,
  conceptSet,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetDate = "event_start_date",
  order = "first",
  inObservation = TRUE,
  nameStyle = "{concept_name}_{window_name}",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
conceptSet	Concept set list.
indexDate	Variable in x that contains the date to compute the intersection.
censorDate	whether to censor overlap events at a date column of x
window	window to consider events in.
targetDate	Event date to use for the intersection.
order	last or first date to use for date/days calculations.
inObservation	If TRUE only records inside an observation period will be considered.
nameStyle	naming of the added column or columns, should include required parameters.
name	Name of the new table, if NULL a temporary table is returned.

Value

table with added columns with overlap information

Examples

```
library(PatientProfiles)
cdm <- mockPatientProfiles()
concept <- dplyr::tibble(
  concept_id = c(1125315),
  domain_id = "Drug",
  vocabulary_id = NA_character_,
  concept_class_id = "Ingredient",
  standard_concept = "S",
  concept_code = NA_character_,
  valid_start_date = as.Date("1900-01-01"),
  valid_end_date = as.Date("2099-01-01"),
  invalid_reason = NA_character_
) %>%
  dplyr::mutate(concept_name = paste0("concept: ", .data$concept_id))
cdm <- CDMConnector::insertTable(cdm, "concept", concept)
result <- cdm$cohort1 %>%
  addConceptIntersectDate(
    conceptSet = list("acetaminophen" = 1125315)
  ) %>%
  dplyr::collect()
mockDisconnect(cdm = cdm)
```

addConceptIntersectDays *It creates column to indicate the days of difference from an index date to a concept*

Description

It creates column to indicate the days of difference from an index date to a concept

Usage

```
addConceptIntersectDays(
  x,
  conceptSet,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetDate = "event_start_date",
  order = "first",
  inObservation = TRUE,
  nameStyle = "{concept_name}_{window_name}",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
conceptSet	Concept set list.
indexDate	Variable in x that contains the date to compute the intersection.
censorDate	whether to censor overlap events at a date column of x
window	window to consider events in.
targetDate	Event date to use for the intersection.
order	last or first date to use for date/days calculations.
inObservation	If TRUE only records inside an observation period will be considered.
nameStyle	naming of the added column or columns, should include required parameters.
name	Name of the new table, if NULL a temporary table is returned.

Value

table with added columns with overlap information

Examples

```
library(PatientProfiles)
cdm <- mockPatientProfiles()
concept <- dplyr::tibble(
  concept_id = c(1125315),
  domain_id = "Drug",
  vocabulary_id = NA_character_,
  concept_class_id = "Ingredient",
  standard_concept = "S",
  concept_code = NA_character_,
  valid_start_date = as.Date("1900-01-01"),
  valid_end_date = as.Date("2099-01-01"),
  invalid_reason = NA_character_
) %>%
  dplyr::mutate(concept_name = paste0("concept: ", .data$concept_id))
cdm <- CDMConnector::insertTable(cdm, "concept", concept)
result <- cdm$cohort1 %>%
  addConceptIntersectDays(
    conceptSet = list("acetaminophen" = 1125315)
  ) %>%
  dplyr::collect()
mockDisconnect(cdm = cdm)
```

---

addConceptIntersectFlag

It creates column to indicate the flag overlap information between a table and a concept

---

Description

It creates column to indicate the flag overlap information between a table and a concept

Usage

```
addConceptIntersectFlag(
  x,
  conceptSet,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetStartDate = "event_start_date",
  targetEndDate = "event_end_date",
  inObservation = TRUE,
  nameStyle = "{concept_name}_{window_name}",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
conceptSet	Concept set list.
indexDate	Variable in x that contains the date to compute the intersection.
censorDate	whether to censor overlap events at a date column of x
window	window to consider events in.
targetStartDate	Event start date to use for the intersection.
targetEndDate	Event end date to use for the intersection.
inObservation	If TRUE only records inside an observation period will be considered.
nameStyle	naming of the added column or columns, should include required parameters.
name	Name of the new table, if NULL a temporary table is returned.

Value

table with added columns with overlap information

Examples

```
library(PatientProfiles)
cdm <- mockPatientProfiles()
concept <- dplyr::tibble(
  concept_id = c(1125315),
  domain_id = "Drug",
  vocabulary_id = NA_character_,
  concept_class_id = "Ingredient",
  standard_concept = "S",
  concept_code = NA_character_,
  valid_start_date = as.Date("1900-01-01"),
  valid_end_date = as.Date("2099-01-01"),
  invalid_reason = NA_character_
) %>%
  dplyr::mutate(concept_name = paste0("concept: ", .data$concept_id))
cdm <- CDMConnector::insertTable(cdm, "concept", concept)
result <- cdm$cohort1 %>%
  addConceptIntersectFlag(
    conceptSet = list("acetaminophen" = 1125315)
  ) %>%
  dplyr::collect()
mockDisconnect(cdm = cdm)
```

addDateOfBirth *Add a column with the individual birth date*

Description

Add a column with the individual birth date

Usage

```
addDateOfBirth(
  x,
  dateOfBirthName = "date_of_birth",
  missingDay = 1,
  missingMonth = 1,
  imposeDay = FALSE,
  imposeMonth = FALSE,
  name = NULL
)
```

Arguments

x	Table in the cdm that contains 'person_id' or 'subject_id'.
dateOfBirthName	Name of the column to be added with the date of birth.
missingDay	Day of the individuals with no or imposed day of birth.
missingMonth	Month of the individuals with no or imposed month of birth.
imposeDay	Whether to impose day of birth.
imposeMonth	Whether to impose month of birth.
name	Name of the new table, if NULL a temporary table is returned.

Value

The function returns the table x with an extra column that contains the date of birth.

Examples

```
library(PatientProfiles)
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addDateOfBirth()
mockDisconnect(cdm = cdm)
```

addDateOfBirthQuery *Query to add a column with the individual birth date*

Description

‘r lifecycle::badge("experimental")‘ Same as ‘addDateOfBirth()‘, except query is not computed to a table.

Usage

```
addDateOfBirthQuery(
  x,
  dateOfBirthName = "date_of_birth",
  missingDay = 1,
  missingMonth = 1,
  imposeDay = FALSE,
  imposeMonth = FALSE
)
```

Arguments

x	Table in the cdm that contains 'person_id' or 'subject_id'.
dateOfBirthName	Name of the column to be added with the date of birth.
missingDay	Day of the individuals with no or imposed day of birth.
missingMonth	Month of the individuals with no or imposed month of birth.
imposeDay	Whether to impose day of birth.
imposeMonth	Whether to impose month of birth.

Value

The function returns the table x with an extra column that contains the date of birth.

Examples

```
library(PatientProfiles)
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addDateOfBirthQuery()
mockDisconnect(cdm = cdm)
```

addDeathDate *Add date of death for individuals. Only death within the same observation period than 'indexDate' will be observed.*

Description

Add date of death for individuals. Only death within the same observation period than 'indexDate' will be observed.

Usage

```
addDeathDate(
  x,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = c(0, Inf),
  deathDateName = "date_of_death",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the window origin.
censorDate	Name of a column to stop followup.
window	window to consider events over.
deathDateName	name of the new column to be added.
name	Name of the new table, if NULL a temporary table is returned.

Value

table x with the added column with death information added.

Examples

```
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addDeathDate()
mockDisconnect(cdm = cdm)
```

addDeathDays *Add days to death for individuals. Only death within the same observation period than 'indexDate' will be observed.*

Description

Add days to death for individuals. Only death within the same observation period than 'indexDate' will be observed.

Usage

```
addDeathDays(
  x,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = c(0, Inf),
  deathDaysName = "days_to_death",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the window origin.
censorDate	Name of a column to stop followup.
window	window to consider events over.
deathDaysName	name of the new column to be added.
name	Name of the new table, if NULL a temporary table is returned.

Value

table x with the added column with death information added.

Examples

```
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addDeathDays()
mockDisconnect(cdm = cdm)
```

addDeathFlag *Add flag for death for individuals. Only death within the same observation period than 'indexDate' will be observed.*

Description

Add flag for death for individuals. Only death within the same observation period than 'indexDate' will be observed.

Usage

```
addDeathFlag(
  x,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = c(0, Inf),
  deathFlagName = "death",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the window origin.
censorDate	Name of a column to stop followup.
window	window to consider events over.
deathFlagName	name of the new column to be added.
name	Name of the new table, if NULL a temporary table is returned.

Value

table x with the added column with death information added.

Examples

```
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addDeathFlag()
mockDisconnect(cdm = cdm)
```

addDemographics *Compute demographic characteristics at a certain date*

Description

Compute demographic characteristics at a certain date

Usage

```
addDemographics(
  x,
  indexDate = "cohort_start_date",
  age = TRUE,
  ageName = "age",
  ageMissingMonth = 1,
  ageMissingDay = 1,
  ageImposeMonth = FALSE,
  ageImposeDay = FALSE,
  ageGroup = NULL,
  missingAgeGroupValue = "None",
  sex = TRUE,
  sexName = "sex",
  missingSexValue = "None",
  priorObservation = TRUE,
  priorObservationName = "prior_observation",
  priorObservationType = "days",
  futureObservation = TRUE,
  futureObservationName = "future_observation",
  futureObservationType = "days",
  dateOfBirth = FALSE,
  dateOfBirthName = "date_of_birth",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the date to compute the demographics characteristics.
age	TRUE or FALSE. If TRUE, age will be calculated relative to indexDate.
ageName	Age variable name.
ageMissingMonth	Month of the year assigned to individuals with missing month of birth.
ageMissingDay	day of the month assigned to individuals with missing day of birth.
ageImposeMonth	TRUE or FALSE. Whether the month of the date of birth will be considered as missing for all the individuals.

`ageImposeDay` TRUE or FALSE. Whether the day of the date of birth will be considered as missing for all the individuals.

`ageGroup` if not NULL, a list of ageGroup vectors.

`missingAgeGroupValue`
Value to include if missing age.

`sex` TRUE or FALSE. If TRUE, sex will be identified.

`sexName` Sex variable name.

`missingSexValue`
Value to include if missing sex.

`priorObservation`
TRUE or FALSE. If TRUE, days of between the start of the current observation period and the indexDate will be calculated.

`priorObservationName`
Prior observation variable name.

`priorObservationType`
Whether to return a "date" or the number of "days".

`futureObservation`
TRUE or FALSE. If TRUE, days between the indexDate and the end of the current observation period will be calculated.

`futureObservationName`
Future observation variable name.

`futureObservationType`
Whether to return a "date" or the number of "days".

`dateOfBirth` TRUE or FALSE, if true the date of birth will be return.

`dateOfBirthName`
dateOfBirth column name.

`name` Name of the new table, if NULL a temporary table is returned.

Value

cohort table with the added demographic information columns.

Examples

```
library(PatientProfiles)
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addDemographics()
mockDisconnect(cdm = cdm)
```

addDemographicsQuery *Query to add demographic characteristics at a certain date*

Description

‘r lifecycle::badge("experimental")‘ Same as ‘addDemographics()‘, except query is not computed to a table.

Usage

```
addDemographicsQuery(
  x,
  indexDate = "cohort_start_date",
  age = TRUE,
  ageName = "age",
  ageMissingMonth = 1,
  ageMissingDay = 1,
  ageImposeMonth = FALSE,
  ageImposeDay = FALSE,
  ageGroup = NULL,
  missingAgeGroupValue = "None",
  sex = TRUE,
  sexName = "sex",
  missingSexValue = "None",
  priorObservation = TRUE,
  priorObservationName = "prior_observation",
  priorObservationType = "days",
  futureObservation = TRUE,
  futureObservationName = "future_observation",
  futureObservationType = "days",
  dateOfBirth = FALSE,
  dateOfBirthName = "date_of_birth"
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the date to compute the demographics characteristics.
age	TRUE or FALSE. If TRUE, age will be calculated relative to indexDate.
ageName	Age variable name.
ageMissingMonth	Month of the year assigned to individuals with missing month of birth.
ageMissingDay	day of the month assigned to individuals with missing day of birth.
ageImposeMonth	TRUE or FALSE. Whether the month of the date of birth will be considered as missing for all the individuals.

`ageImposeDay` TRUE or FALSE. Whether the day of the date of birth will be considered as missing for all the individuals.

`ageGroup` if not NULL, a list of ageGroup vectors.

`missingAgeGroupValue`
Value to include if missing age.

`sex` TRUE or FALSE. If TRUE, sex will be identified.

`sexName` Sex variable name.

`missingSexValue`
Value to include if missing sex.

`priorObservation`
TRUE or FALSE. If TRUE, days of between the start of the current observation period and the indexDate will be calculated.

`priorObservationName`
Prior observation variable name.

`priorObservationType`
Whether to return a "date" or the number of "days".

`futureObservation`
TRUE or FALSE. If TRUE, days between the indexDate and the end of the current observation period will be calculated.

`futureObservationName`
Future observation variable name.

`futureObservationType`
Whether to return a "date" or the number of "days".

`dateOfBirth` TRUE or FALSE, if true the date of birth will be return.

`dateOfBirthName`
dateOfBirth column name.

Value

cohort table with the added demographic information columns.

Examples

```
library(PatientProfiles)
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addDemographicsQuery()
mockDisconnect(cdm = cdm)
```

addFutureObservation *Compute the number of days till the end of the observation period at a certain date*

Description

Compute the number of days till the end of the observation period at a certain date

Usage

```
addFutureObservation(
  x,
  indexDate = "cohort_start_date",
  futureObservationName = "future_observation",
  futureObservationType = "days",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the date to compute the future observation.
futureObservationName	name of the new column to be added.
futureObservationType	Whether to return a "date" or the number of "days".
name	Name of the new table, if NULL a temporary table is returned.

Value

cohort table with added column containing future observation of the individuals.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addFutureObservation()
mockDisconnect(cdm = cdm)
```

addFutureObservationQuery *Query to add the number of days till the end of the observation period at a certain date*

Description

‘r lifecycle::badge("experimental")‘ Same as ‘addFutureObservation()‘, except query is not computed to a table.

Usage

```
addFutureObservationQuery(
  x,
  indexDate = "cohort_start_date",
  futureObservationName = "future_observation",
  futureObservationType = "days"
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the date to compute the future observation.
futureObservationName	name of the new column to be added.
futureObservationType	Whether to return a "date" or the number of "days".

Value

cohort table with added column containing future observation of the individuals.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addFutureObservationQuery()
mockDisconnect(cdm = cdm)
```

addInObservation *Indicate if a certain record is within the observation period*

Description

Indicate if a certain record is within the observation period

Usage

```
addInObservation(
  x,
  indexDate = "cohort_start_date",
  window = c(0, 0),
  completeInterval = FALSE,
  nameStyle = "in_observation",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the date to compute the observation flag.
window	window to consider events of.
completeInterval	If the individuals are in observation for the full window.
nameStyle	Name of the new columns to create, it must contain "window_name" if multiple windows are provided.
name	Name of the new table, if NULL a temporary table is returned.

Value

cohort table with the added binary column assessing inObservation.

Examples

```
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addInObservation()
mockDisconnect(cdm = cdm)
```

addInObservationQuery *Query to add a new column to indicate if a certain record is within the observation period*

Description

‘r lifecycle::badge("experimental")‘ Same as ‘addInObservation()‘, except query is not computed to a table.

Usage

```
addInObservationQuery(
  x,
  indexDate = "cohort_start_date",
  window = c(0, 0),
  completeInterval = FALSE,
  nameStyle = "in_observation"
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the date to compute the observation flag.
window	window to consider events of.
completeInterval	If the individuals are in observation for the full window.
nameStyle	Name of the new columns to create, it must contain "window_name" if multiple windows are provided.

Value

cohort table with the added binary column assessing inObservation.

Examples

```
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addInObservationQuery()
mockDisconnect(cdm = cdm)
```

addObservationPeriodId *Add the ordinal number of the observation period associated that a given date is in.*

Description

Add the ordinal number of the observation period associated that a given date is in.

Usage

```
addObservationPeriodId(
  x,
  indexDate = "cohort_start_date",
  nameObservationPeriodId = "observation_period_id",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the date to compute the observation flag.
nameObservationPeriodId	Name of the new colum.
name	Name of the new table, if NULL a temporary table is returned.

Value

Table with the current observation period id added.

Examples

```
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addObservationPeriodId()
mockDisconnect(cdm = cdm)
```

addObservationPeriodIdQuery *Add the ordinal number of the observation period associated that a given date is in. Result is not computed, only query is added.*

Description

Add the ordinal number of the observation period associated that a given date is in. Result is not computed, only query is added.

Usage

```
addObservationPeriodIdQuery(
  x,
  indexDate = "cohort_start_date",
  nameObservationPeriodId = "observation_period_id"
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the date to compute the observation flag.
nameObservationPeriodId	Name of the new colum.

Value

Table with the current observation period id added.

Examples

```
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addObservationPeriodIdQuery()
mockDisconnect(cdm = cdm)
```

addPriorObservation *Compute the number of days of prior observation in the current observation period at a certain date*

Description

Compute the number of days of prior observation in the current observation period at a certain date

Usage

```
addPriorObservation(
  x,
  indexDate = "cohort_start_date",
  priorObservationName = "prior_observation",
  priorObservationType = "days",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the date to compute the prior observation.
priorObservationName	name of the new column to be added.
priorObservationType	Whether to return a "date" or the number of "days".
name	Name of the new table, if NULL a temporary table is returned.

Value

cohort table with added column containing prior observation of the individuals.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addPriorObservation()
mockDisconnect(cdm = cdm)
```

addPriorObservationQuery *Query to add the number of days of prior observation in the current observation period at a certain date*

Description

‘r lifecycle::badge("experimental")‘ Same as ‘addPriorObservation()‘, except query is not computed to a table.

Usage

```
addPriorObservationQuery(
  x,
  indexDate = "cohort_start_date",
  priorObservationName = "prior_observation",
  priorObservationType = "days"
)
```

Arguments

x	Table with individuals in the cdm.
indexDate	Variable in x that contains the date to compute the prior observation.
priorObservationName	name of the new column to be added.
priorObservationType	Whether to return a "date" or the number of "days".

Value

cohort table with added column containing prior observation of the individuals.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addPriorObservationQuery()
mockDisconnect(cdm = cdm)
```

addSex *Compute the sex of the individuals*

Description

Compute the sex of the individuals

Usage

```
addSex(x, sexName = "sex", missingSexValue = "None", name = NULL)
```

Arguments

x	Table with individuals in the cdm.
sexName	name of the new column to be added.
missingSexValue	Value to include if missing sex.
name	Name of the new table, if NULL a temporary table is returned.

Value

table x with the added column with sex information.

Examples

```
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addSex()
mockDisconnect(cdm = cdm)
```

addSexQuery *Query to add the sex of the individuals*

Description

‘r lifecycle::badge("experimental")‘ Same as ‘addSex()‘, except query is not computed to a table.

Usage

```
addSexQuery(x, sexName = "sex", missingSexValue = "None")
```

Arguments

x	Table with individuals in the cdm.
sexName	name of the new column to be added.
missingSexValue	Value to include if missing sex.

Value

table x with the added column with sex information.

Examples

```
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addSexQuery()
mockDisconnect(cdm = cdm)
```

---

addTableIntersectCount

Compute number of intersect with an omop table.

Description

Compute number of intersect with an omop table.

Usage

```
addTableIntersectCount(
  x,
  tableName,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetStartDate = startDateColumn(tableName),
  targetEndDate = endDateColumn(tableName),
  nameStyle = "{table_name}_{window_name}",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
tableName	Name of the table to intersect with. Options: visit_occurrence, condition_occurrence, drug_exposure, procedure_occurrence, device_exposure, measurement, observation, drug_era, condition_era, specimen, episode.

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.
window	window to consider events in.
targetStartDate	Column name with start date for comparison.
targetEndDate	Column name with end date for comparison.
nameStyle	naming of the added column or columns, should include required parameters.
name	Name of the new table, if NULL a temporary table is returned.

Value

table with added columns with intersect information.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addTableIntersectCount(tableName = "visit_occurrence")

mockDisconnect(cdm = cdm)
```

---

addTableIntersectDate *Compute date of intersect with an omop table.*

Description

Compute date of intersect with an omop table.

Usage

```
addTableIntersectDate(
  x,
  tableName,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetDate = startDateColumn(tableName),
  order = "first",
  nameStyle = "{table_name}_{window_name}",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
tableName	Name of the table to intersect with. Options: visit_occurrence, condition_occurrence, drug_exposure, procedure_occurrence, device_exposure, measurement, observation, drug_era, condition_era, specimen, episode.
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.
window	window to consider events in.
targetDate	Target date in tableName.
order	which record is considered in case of multiple records (only required for date and days options).
nameStyle	naming of the added column or columns, should include required parameters.
name	Name of the new table, if NULL a temporary table is returned.

Value

table with added columns with intersect information.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addTableIntersectDate(tableName = "visit_occurrence")

mockDisconnect(cdm = cdm)
```

---

addTableIntersectDays *Compute time to intersect with an omop table.*

Description

Compute time to intersect with an omop table.

Usage

```
addTableIntersectDays(
  x,
  tableName,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetDate = startDateColumn(tableName),
```

```
  order = "first",
  nameStyle = "{table_name}_{window_name}",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
tableName	Name of the table to intersect with. Options: visit_occurrence, condition_occurrence, drug_exposure, procedure_occurrence, device_exposure, measurement, observation, drug_era, condition_era, specimen, episode.
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.
window	window to consider events in.
targetDate	Target date in tableName.
order	which record is considered in case of multiple records (only required for date and days options).
nameStyle	naming of the added column or columns, should include required parameters.
name	Name of the new table, if NULL a temporary table is returned.

Value

table with added columns with intersect information.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addTableIntersectDays(tableName = "visit_occurrence")

mockDisconnect(cdm = cdm)
```

---

addTableIntersectField

Intersecting the cohort with columns of an OMOP table of user's choice. It will add an extra column to the cohort, indicating the intersected entries with the target columns in a window of the user's choice.

Description

Intersecting the cohort with columns of an OMOP table of user's choice. It will add an extra column to the cohort, indicating the intersected entries with the target columns in a window of the user's choice.

Usage

```
addTableIntersectField(
  x,
  tableName,
  field,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetDate = startDateColumn(tableName),
  order = "first",
  nameStyle = "{table_name}_{extra_value}_{window_name}",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
tableName	Name of the table to intersect with. Options: visit_occurrence, condition_occurrence, drug_exposure, procedure_occurrence, device_exposure, measurement, observation, drug_era, condition_era, specimen, episode.
field	The columns from the table in tableName to intersect over. For example, if the user uses visit_occurrence in tableName then for field the possible options include visit_occurrence_id, visit_concept_id, visit_type_concept_id.
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.
window	window to consider events in when intersecting with the chosen column.
targetDate	The dates in the target columns in tableName that the user may want to restrict to.
order	which record is considered in case of multiple records (only required for date and days options).
nameStyle	naming of the added column or columns, should include required parameters.
name	Name of the new table, if NULL a temporary table is returned.

Value

table with added columns with intersect information.

Examples

```
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addTableIntersectField(
    tableName = "visit_occurrence",
    field = "visit_concept_id",
    order = "last",
    window = c(-Inf, -1)
```

```
  )
mockDisconnect(cdm = cdm)
```

---

addTableIntersectFlag *Compute a flag intersect with an omop table.*

Description

Compute a flag intersect with an omop table.

Usage

```
addTableIntersectFlag(
  x,
  tableName,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetStartDate = startDateColumn(tableName),
  targetEndDate = endDateColumn(tableName),
  nameStyle = "{table_name}_{window_name}",
  name = NULL
)
```

Arguments

x	Table with individuals in the cdm.
tableName	Name of the table to intersect with. Options: visit_occurrence, condition_occurrence, drug_exposure, procedure_occurrence, device_exposure, measurement, observation, drug_era, condition_era, specimen, episode.
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.
window	window to consider events in.
targetStartDate	Column name with start date for comparison.
targetEndDate	Column name with end date for comparison.
nameStyle	naming of the added column or columns, should include required parameters.
name	Name of the new table, if NULL a temporary table is returned.

Value

table with added columns with intersect information.

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addTableIntersectFlag(tableName = "visit_occurrence")
mockDisconnect(cdm = cdm)
```

`availableEstimates` *Show the available estimates that can be used for the different variable_type supported.*

Description

Show the available estimates that can be used for the different variable_type supported.

Usage

```
availableEstimates(variableType = NULL, fullQuantiles = FALSE)
```

Arguments

`variableType`	A set of variable types.
`fullQuantiles`	Whether to display the exact quantiles that can be computed or only the qXX to summarise all of them.

Value

A tibble with the available estimates.

Examples

```
library(PatientProfiles)

availableEstimates()
availableEstimates("numeric")
availableEstimates(c("numeric", "categorical"))
```

endDateColumn *Get the name of the end date column for a certain table in the cdm*

Description

Get the name of the end date column for a certain table in the cdm

Usage

```
endDateColumn(tableName)
```

Arguments

tableName	Name of the table.

Value

Name of the end date column in that table.

Examples

```
library(PatientProfiles)
endDateColumn("condition_occurrence")
```

mockDisconnect *Function to disconnect from the mock*

Description

Function to disconnect from the mock

Usage

```
mockDisconnect(cdm)
```

Arguments

cdm	A cdm_reference object.

`mockPatientProfiles` *It creates a mock database for testing PatientProfiles package*

Description

It creates a mock database for testing PatientProfiles package

Usage

```
mockPatientProfiles(
  con = NULL,
  writeSchema = NULL,
  numberIndividuals = 10,
  ...,
  seed = NULL
)
```

Arguments

`con`	A DBI connection to create the cdm mock object.
`writeSchema`	Name of an schema on the same connection with writing permisions.
`numberIndividuals`	Number of individuals to create in the cdm reference.
`...`	User self defined tables to put in cdm, it can input as many as the user want.
`seed`	A number to set the seed. If NULL seed is not used.

Value

A mock cdm_reference object created following user's specifications.

Examples

```
library(PatientProfiles)
library(CDMConnector)

cdm <- mockPatientProfiles()

mockDisconnect(cdm = cdm)
```

sourceConceptIdColumn *Get the name of the source concept_id column for a certain table in the cdm*

Description

Get the name of the source concept_id column for a certain table in the cdm

Usage

```
sourceConceptIdColumn(tableName)
```

Arguments

tableName	Name of the table.

Value

Name of the source_concept_id column in that table.

Examples

```
library(PatientProfiles)
sourceConceptIdColumn("condition_occurrence")
```

standardConceptIdColumn *Get the name of the standard concept_id column for a certain table in the cdm*

Description

Get the name of the standard concept_id column for a certain table in the cdm

Usage

```
standardConceptIdColumn(tableName)
```

Arguments

tableName	Name of the table.

Value

Name of the concept_id column in that table.

Examples

```
library(PatientProfiles)
standardConceptIdColumn("condition_occurrence")
```

`startDateColumn` *Get the name of the start date column for a certain table in the cdm*

Description

Get the name of the start date column for a certain table in the cdm

Usage

```
startDateColumn(tableName)
```

Arguments

`tableName`	Name of the table.

Value

Name of the start date column in that table.

Examples

```
library(PatientProfiles)
startDateColumn("condition_occurrence")
```

`summariseResult` *Summarise variables using a set of estimate functions. The output will be a formatted summarised_result object.*

Description

Summarise variables using a set of estimate functions. The output will be a formatted summarised_result object.

Usage

```
summariseResult(
  table,
  group = list(),
  includeOverallGroup = FALSE,
  strata = list(),
  includeOverallStrata = TRUE,
  variables = NULL,
  estimates = c("min", "q25", "median", "q75", "max", "count", "percentage"),
  counts = TRUE
)
```

Arguments

table	Table with different records.
group	List of groups to be considered.
includeOverallGroup	TRUE or FALSE. If TRUE, results for an overall group will be reported when a list of groups has been specified.
strata	List of the stratifications within each group to be considered.
includeOverallStrata	TRUE or FALSE. If TRUE, results for an overall strata will be reported when a list of strata has been specified.
variables	Variables to summarise, it can be a list to point to different set of estimate names.
estimates	Estimates to obtain, it can be a list to point to different set of variables.
counts	Whether to compute number of records and number of subjects.

Value

A summarised_result object with the summarised data of interest.

Examples

```
library(PatientProfiles)
library(dplyr)

cdm <- mockPatientProfiles()
x <- cdm$cohort1 %>%
  addDemographics() %>%
  collect()
result <- summariseResult(x)
mockDisconnect(cdm = cdm)
```

`variableTypes` *Classify the variables between 5 types: "numeric", "categorical", "binary", "date", or NA.*

Description

Classify the variables between 5 types: "numeric", "categorical", "binary", "date", or NA.

Usage

```
variableTypes(table)
```

Arguments

`table`	Tibble.

Value

Tibble with the variables type and classification.

Examples

```
library(PatientProfiles)
x <- dplyr::tibble(
  person_id = c(1, 2),
  start_date = as.Date(c("2020-05-02", "2021-11-19")),
  asthma = c(0, 1)
)
variableTypes(x)
```

Index