Package ‘PatientProfiles’

June 14, 2023

Type Package

Title Identify Characteristics of Patients in the OMOP Common Data Model

Version 0.2.0

Maintainer Marti Catala <marti.catalasabate@ndoms.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 7.2.3

Suggests covr, duckdb, testthat (>= 3.1.5), knitr, CodelistGenerator, rmarkdown, glue, stringr, odbc, ggplot2, spelling, RPostgres, dbplyr

Imports magrittr, CDMConnector (>= 1.0.0), dplyr, tidyr, checkmate, lubridate, DBI, rlang, cli, pillar

VignetteBuilder knitr

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

Language en-US

Depends R (>= 2.10)

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 [ctb] (<https://orcid.org/0009-0006-7948-3747>), Xintong Li [ctb] (<https://orcid.org/0000-0002-6872-5804>)

Repository CRAN

Date/Publication 2023-06-14 03:50:02 UTC
R topics documented:

addAge ........................................... 2
addAttributes ................................. 4
addCategories .................................. 5
addCohortIntersect ......................... 6
addCohortIntersectCount ................. 9
addCohortIntersectDate ................... 11
addCohortIntersectDays ................. 13
addCohortIntersectFlag ................. 15
addDateOfBirth ............................... 18
addDemographics ............................ 19
addFutureObservation ..................... 20
addInObservation ......................... 22
addIntersect .................................. 23
addPriorHistory ............................. 24
addSex ........................................ 26
availableFunctions ......................... 27
detectVariables .............................. 27
getConceptName ................................ 28
getEndName .................................... 29
getStartName .................................. 29
mockPatientProfiles ....................... 30
summariseCharacteristics .............. 33
summariseResult ............................. 34
variableTypes ............................... 35

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,
  cdm,
  indexDate = "cohort_start_date",
  ageName = "age",
  ageGroup = NULL,
  ageDefaultMonth = 1,
  ageDefaultDay = 1,
  ageImposeMonth = FALSE,
  ageImposeDay = FALSE,
addAge

    tablePrefix = NULL

Arguments

x                Table with individuals in the cdm.
cdm             Object that contains a cdm reference. Use CDMConnector to obtain a cdm reference.
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.
ageDefaultMonth Month of the year assigned to individuals with missing month of birth. By default: 1.
ageDefaultDay   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.
tablePrefix     The stem for the permanent tables that will be created. If NULL, temporary tables will be used throughout.

Value

tibble with the age column added

Examples

library(DBI)
library(duckdb)
library(PatientProfiles)
cohort1 <- dplyr::tibble(
    cohort_definition_id = c("1", "1", "1"),
    subject_id = c("1", "2", "3"),
    cohort_start_date = c(
        as.Date("2010-01-01"), as.Date("2010-01-01"), as.Date("2010-01-01")
    ),
    cohort_end_date = c(
        as.Date("2015-01-01"), as.Date("2013-01-01"), as.Date("2018-01-01")
    )
)

person <- dplyr::tibble(
    person_id = c("1", "2", "3"),
    gender_concept_id = c("8507", "8532", "8507"),
addAttributes

Get attributes from one cohort to another

Description
Get attributes from one cohort to another

Usage
addAttributes(newcohort, oldcohort)

Arguments
newcohort cohort to which to attach the attributes
oldcohort cohort from which to get the attributes

Value
new cohort with added attributes from the other given cohort

Examples

```r
library(CDMConnector)
library(PatientProfiles)
library(dplyr)

cdm <- mockPatientProfiles()
attributes(cdm$cohort1)
x <- cdm$cohort1 %>%
  filter(cohort_definition_id == 1) %>%
  computeQuery()
attributes(x)
x <- addAttributes(x, cdm$cohort1)
attributes(cdm$cohort1)
```
addCategories  Categorize a numeric variable

Description

Categorize a numeric variable

Usage

addCategories(
  x,
  cdm,
  variable,
  categories,
  missingCategoryValue = "None",
  overlap = FALSE,
  tablePrefix = NULL
)

Arguments

x  Table with individuals in the cdm

cdm  Object that contains a cdm reference. Use CDMConnector to obtain a cdm reference.

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

tablePrefix  The stem for the permanent tables that will be created. If NULL, temporary tables will be used throughout.

Value

tibble with the categorical variable added.

Examples

#'  

library(DBI)
library(duckdb)
library(PatientProfiles)
cohort1 <- dplyr::tibble(  
  cohort_definition_id = c("1", "1", "1"),  
  cohort_definition_id = c("1", "1", "1"),
subject_id = c("1", "2", "3"),
cohort_start_date = c(
  as.Date("2010-03-03"), as.Date("2010-03-01"), as.Date("2010-02-01"))
),
cohort_end_date = c(
  as.Date("2015-01-01"), as.Date("2013-01-01"), as.Date("2013-01-01"))
)
person <- dplyr::tibble(
  person_id = c("1", "2", "3"),
gender_concept_id = c("8507", "8507", "8507"),
month_of_birth = c(03, 07, NA),
day_of_birth = c(NA, 02, 01)
)
cdm <- mockPatientProfiles(person = person, cohort1 = cohort1)
result <- cdm$cohort1 %>%
  addAge(cdm) %>%
  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)
    ))
  )

addCohortIntersect  
Compute the intersect with a target cohort, you can compute the number of occurrences, a flag of presence, a certain date and/or the time difference

Description

Compute the intersect with a target cohort, you can compute the number of occurrences, a flag of presence, a certain date and/or the time difference

Usage

addCohortIntersect(
  x,
  cdm,
  targetCohortTable,
  targetCohortId = NULL,
  indexDate = "cohort_start_date",
  targetStartDate = "cohort_start_date",
  targetEndDate = "cohort_end_date",
)
addCohortIntersect

```r
  window = list(c(0, Inf)),
  order = "first",
  flag = TRUE,
  count = TRUE,
  date = TRUE,
  days = TRUE,
  nameStyle = "{value}_{cohort_name}_{window_name}",
  tablePrefix = NULL
)
```

**Arguments**

- `x`: Table with individuals in the cdm
- `cdm`: Object that contains a cdm reference. Use CDMConnector to obtain a cdm reference.
- `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.
- `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
- `order`: which record is considered in case of multiple records
- `flag`: TRUE or FALSE. If TRUE, flag will calculated for this intersection
- `count`: TRUE or FALSE. If TRUE, the number of counts will be calculated for this intersection
- `date`: TRUE or FALSE. If TRUE, date will be calculated for this intersection
- `days`: TRUE or FALSE. If TRUE, time difference in days will be calculated for this intersection
- `nameStyle`: naming of the added column or columns, should include required parameters
- `tablePrefix`: The stem for the permanent tables that will be created. If NULL, temporary tables will be used throughout.

**Value**

table with added columns with overlap information

**Examples**

```r
  cohort1 <- dplyr::tibble(
    cohort_definition_id = c(1, 1, 1, 1, 1),
    subject_id = c(1, 1, 1, 2, 2),
    cohort_start_date = as.Date(
```
addCohortIntersect

c(
    "2020-01-01",
    "2020-01-15",
    "2020-01-20",
    "2020-01-01",
    "2020-02-01"
)
),
cohort_end_date = as.Date(
    c(
        "2020-01-01",
        "2020-01-15",
        "2020-01-20",
        "2020-01-01",
        "2020-02-01"
    )
)
)
}

cohort2 <- dplyr::tibble(
    cohort_definition_id = c(1, 1, 1, 1, 1, 1, 1),
    subject_id = c(1, 1, 1, 2, 2, 2, 1),
    cohort_start_date = as.Date(
        c(
            "2020-01-15",
            "2020-01-25",
            "2020-01-26",
            "2020-01-29",
            "2020-03-15",
            "2020-01-24",
            "2020-02-16"
        )
    ),
    cohort_end_date = as.Date(
        c(
            "2020-01-15",
            "2020-01-25",
            "2020-01-26",
            "2020-01-29",
            "2020-03-15",
            "2020-01-24",
            "2020-02-16"
        )
    ),
)

cdm <- mockPatientProfiles(cohort1 = cohort1, cohort2 = cohort2)

result <- cdm$cohort1 %>%
    addCohortIntersect(
        cdm = cdm,
        targetCohortTable = "cohort2"
    ) %>%
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,
  cdm,
  targetCohortTable,
  targetCohortId = NULL,
  indexDate = "cohort_start_date",
  targetStartDate = "cohort_start_date",
  targetEndDate = "cohort_end_date",
  window = list(c(0, Inf)),
  nameStyle = "{cohort_name}_{window_name}",
  tablePrefix = NULL
)

Arguments

x Table with individuals in the cdm
cdm Object that contains a cdm reference. Use CDMConnector to obtain a cdm reference.
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.
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
tablePrefix The stem for the permanent tables that will be created. If NULL, temporary tables will be used throughout.
Value

table with added columns with overlap information

Examples

library(PatientProfiles)
library(dplyr)

cohort1 <- dplyr::tibble(
  cohort_definition_id = c(1, 1, 1, 1, 1),
  subject_id = c(1, 1, 1, 2, 2),
  cohort_start_date = as.Date(c(
    "2020-01-01",
    "2020-01-15",
    "2020-01-20",
    "2020-01-01",
    "2020-02-01"
  )),
  cohort_end_date = as.Date(c(
    "2020-01-01",
    "2020-01-15",
    "2020-01-20",
    "2020-01-01",
    "2020-02-01"
  ))
)

cohort2 <- dplyr::tibble(
  cohort_definition_id = c(1, 1, 1, 1, 1, 1, 1),
  subject_id = c(1, 1, 1, 2, 2, 2, 1),
  cohort_start_date = as.Date(c(
    "2020-01-15",
    "2020-01-25",
    "2020-01-26",
    "2020-01-29",
    "2020-03-15",
    "2020-01-24",
    "2020-02-16"
  )),
  cohort_end_date = as.Date(c(
    "2020-01-15",
    "2020-01-25",
    "2020-01-26",
    "2020-01-29",
    "2020-01-29",
    "2020-01-26",
    "2020-01-29"
  ))
)
addCohortIntersectDate

“2020-03-15”,
“2020-01-24”,
“2020-02-16”
)
)

cdm <- mockPatientProfiles(cohort1 = cohort1, cohort2 = cohort2)

result <- cdm$cohort1 %>%
  addCohortIntersectCount(
    cdm = cdm,
    targetCohortTable = "cohort2"
  ) %>%
  dplyr::collect()

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,
  cdm,
  targetCohortTable,
  targetCohortId = NULL,
  indexDate = "cohort_start_date",
  targetDate = "cohort_start_date",
  order = "first",
  window = c(0, Inf),
  nameStyle = "{cohort_name}_{window_name}",
  tablePrefix = NULL
)

Arguments

x

Table with individuals in the cdm
cdm

Object that contains a cdm reference. Use CDMConnector to obtain a cdm reference.
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.

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

tablePrefix  The stem for the permanent tables that will be created. If NULL, temporary tables will be used throughout.

Value

x along with additional columns for each cohort of interest.

Examples

library(PatientProfiles)
library(dplyr)

cohort1 <- dplyr::tibble(
  cohort_definition_id = c(1, 1, 1, 1, 1),
  subject_id = c(1, 1, 1, 2, 2),
  cohort_start_date = as.Date(c(
    "2020-01-01",
    "2020-01-15",
    "2020-01-20",
    "2020-01-01",
    "2020-02-01"
  )),
  cohort_end_date = as.Date(c(
    "2020-01-01",
    "2020-01-15",
    "2020-01-20",
    "2020-01-01",
    "2020-02-01"
  )
)

cohort2 <- dplyr::tibble(
  cohort_definition_id = c(1, 1, 1, 1, 1, 1, 1),
  subject_id = c(1, 1, 1, 2, 2, 2, 1),
  cohort_start_date = as.Date(c(
    "2020-01-01",
    "2020-01-15",
    "2020-01-20",
    "2020-01-01",
    "2020-02-01"
  )
)
)
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, 
cdm, 
targetCohortTable, 
targetCohortId = NULL, 
)
addCohortIntersectDays

indexDate = "cohort_start_date",
targetDate = "cohort_start_date",
order = "first",
window = c(0, Inf),
nameStyle = "{cohort_name}_{window_name}"

tablePrefix = NULL
)

Arguments

x Table with individuals in the cdm
cdm Object that contains a cdm reference. Use CDMConnector to obtain a cdm reference.
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.
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
tablePrefix The stem for the permanent tables that will be created. If NULL, temporary tables will be used throughout.

Value

x along with additional columns for each cohort of interest.

Examples

library(PatientProfiles)
library(dplyr)

cohort1 <- dplyr::tibble(
  cohort_definition_id = c(1, 1, 1, 1),
  subject_id = c(1, 1, 1, 2),
  cohort_start_date = as.Date(c(
    "2020-01-01",
    "2020-01-15",
    "2020-01-20",
    "2020-01-01",
    "2020-02-01"
  )
)
addCohortIntersectFlag

),
cohort_end_date = as.Date(
  c(
    "2020-01-01",
    "2020-01-15",
    "2020-01-20",
    "2020-01-01",
    "2020-02-01"
  )
)
)

cohort2 <- dplyr::tibble(
  cohort_definition_id = c(1, 1, 1, 1, 1, 1, 1),
  subject_id = c(1, 1, 1, 2, 2, 2, 1),
  cohort_start_date = as.Date(
    c(
      "2020-01-15",
      "2020-01-25",
      "2020-01-26",
      "2020-01-29",
      "2020-03-15",
      "2020-01-24",
      "2020-02-16"
    )
  ),
  cohort_end_date = as.Date(
    c(
      "2020-01-15",
      "2020-01-25",
      "2020-01-26",
      "2020-01-29",
      "2020-03-15",
      "2020-01-24",
      "2020-02-16"
    )
  ),
)

cdm <- mockPatientProfiles(cohort1 = cohort1, cohort2 = cohort2)

result <- cdm$cohort1 %>%
  addCohortIntersectDays(
    cdm = cdm,
    targetCohortTable = "cohort2"
  ) %>%
dplyr::collect()
addCohortIntersectFlag

It creates columns to indicate the presence of cohorts

Description

It creates columns to indicate the presence of cohorts

Usage

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

Arguments

x Table with individuals in the cdm

cdm Object that contains a cdm reference. Use CDMConnector to obtain a cdm reference.

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.

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

tablePrefix The stem for the permanent tables that will be created. If NULL, temporary tables will be used throughout.

Value

table with added columns with overlap information
Examples

cohort1 <- dplyr::tibble(
    cohort_definition_id = c(1, 1, 1, 1, 1),
    subject_id = c(1, 1, 1, 2, 2),
    cohort_start_date = as.Date(
        c(
            "2020-01-01",
            "2020-01-15",
            "2020-01-20",
            "2020-01-01",
            "2020-02-01"
        )
    ),
    cohort_end_date = as.Date(
        c(
            "2020-01-01",
            "2020-01-15",
            "2020-01-20",
            "2020-01-01",
            "2020-02-01"
        )
    )
)

cohort2 <- dplyr::tibble(
    cohort_definition_id = c(1, 1, 1, 1, 1, 1, 1),
    subject_id = c(1, 1, 1, 2, 2, 2, 1),
    cohort_start_date = as.Date(
        c(
            "2020-01-15",
            "2020-01-25",
            "2020-01-26",
            "2020-01-29",
            "2020-03-15",
            "2020-01-24",
            "2020-02-16"
        )
    ),
    cohort_end_date = as.Date(
        c(
            "2020-01-15",
            "2020-01-25",
            "2020-01-26",
            "2020-01-29",
            "2020-03-15",
            "2020-01-24",
            "2020-02-16"
        )
    )
)
```r
cdm <- mockPatientProfiles(cohort1 = cohort1, cohort2 = cohort2)
result <- cdm$cohort1 %>%
  addCohortIntersectFlag(
    cdm = cdm,
    targetCohortTable = "cohort2"
  ) %>%
  dplyr::collect()
```

---

**addDateOfBirth**  
*Add a column with the individual birth date*

**Description**

Add a column with the individual birth date

**Usage**

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

**Arguments**

- `x`: Table in the cdm that contains `person_id` or `subject_id`  
- `cdm`: `cdm` object created with CDMConnector::cdm_from_con().  
- `name`: 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 %>%
  addDateOfBirth(cdm)

addDemographics

Compute demographic characteristics at a certain date

Description

Compute demographic characteristics at a certain date

Usage

addDemographics(
  x,
  cdm,
  indexDate = "cohort_start_date",
  age = TRUE,
  ageName = "age",
  ageDefaultMonth = 1,
  ageDefaultDay = 1,
  ageImposeMonth = FALSE,
  ageImposeDay = FALSE,
  ageGroup = NULL,
  sex = TRUE,
  sexName = "sex",
  priorHistory = TRUE,
  priorHistoryName = "prior_history",
  futureObservation = TRUE,
  futureObservationName = "future_observation",
  tablePrefix = NULL
)

Arguments

x Table with individuals in the cdm
cdm Object that contains a cdm reference. Use CDMConnector to obtain a cdm reference.
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
addFutureObservation

ageDefaultMonth
   Month of the year assigned to individuals with missing month of birth.

ageDefaultDay
   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

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

sexName
   Sex variable name

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

priorHistoryName
   Prior history variable name

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

tablePrefix
   The stem for the permanent tables that will be created. If NULL, temporary
   tables will be used throughout.

Value
   cohort table with the added demographic information columns

Examples

library(PatientProfiles)
cdm <- mockPatientProfiles()
cdm$cohort1 %>% addDemographics(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
addFutureObservation

Usage

addFutureObservation(
  x,
  cdm,
  indexDate = "cohort_start_date",
  futureObservationName = "future_observation",
  tablePrefix = NULL
)

Arguments

x Table with individuals in the cdm.
cdm Object that contains a cdm reference. Use CDMConnector to obtain a cdm reference.
indexDate Variable in x that contains the date to compute the future observation.
futureObservationName name of the new column to be added
tablePrefix The stem for the permanent tables that will be created. If NULL, temporary tables will be used throughout.

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

Examples

library(DBI)
library(duckdb)
library(PatientProfiles)
cohort1 <- dplyr::tibble(
  cohort_definition_id = c("1", "1", "1"),
  subject_id = c("1", "2", "3"),
  cohort_start_date = c(
    as.Date("2010-03-03"),
    as.Date("2010-03-01"),
    as.Date("2010-02-01")
  ),
  cohort_end_date = c(
    as.Date("2015-01-01"),
    as.Date("2013-01-01"),
    as.Date("2013-01-01")
  )
)
obs_1 <- dplyr::tibble(
  observation_period_id = c("1", "2", "3"),
  person_id = c("1", "2", "3"),
  observation_period_start_date = c(
    ...)
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,
  cdm,
  indexDate = "cohort_start_date",
  name = "in_observation",
  tablePrefix = NULL
)

Arguments

x               Table with individuals in the cdm.
cdm             Object that contains a cdm reference. Use CDMConnector to obtain a cdm reference.
indexDate       Variable in x that contains the date to compute the observation flag.
name             name of the column to hold the result of the query: 1 if the individual is in observation, 0 if not
tablePrefix     The stem for the permanent tables that will be created. If NULL, temporary tables will be used throughout.
**addIntersect**

Value

cohort table with the added binary column assessing inObservation

Examples

```r
library(PatientProfiles)
cdm <- mockPatientProfiles()
cdm$cohort1 %>% addInObservation(cdm)
```

**Description**

It creates columns to indicate overlap information between two tables

**Usage**

```r
addIntersect(
  x, cdm, tableName, value,
  filterVariable = NULL, filterId = NULL, idName = NULL,
  window = list(c(0, Inf)), indexDate = "cohort_start_date",
  targetStartDate = getStartName(tableName), targetEndDate = getEndName(tableName),
  order = "first",
  nameStyle = "\{value\}_{id_name}_{window_name}\",
  tablePrefix = NULL
)
```

**Arguments**

- **x**: Table with individuals in the cdm
- **cdm**: Object that contains a cdm reference. Use CDMConnector to obtain a cdm reference.
- **tableName**: name of the cohort that we want to check for overlap
- **value**: value of interest to add: it can be count, flag, date or time
- **filterVariable**: the variable that we are going to use to filter (e.g. cohort_definition_id)
addPriorHistory

filterId the value of filterVariable that we are interested in, it can be a vector
idName the name of each filterId, must have same length than filterId
window window to consider events of
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)
order last or first date to use for date/time calculations
nameStyle naming of the added column or columns, should include required parameters
tablePrefix The stem for the permanent tables that will be created. If NULL, temporary tables will be used throughout.

Value
table with added columns with overlap information

Examples

library(PatientProfiles)

cdm <- mockPatientProfiles()
result <- cdm$cohort1 %>%
  addIntersect(
    cdm = cdm, tableName = "cohort2", value = "date"
  ) %>%
dplyr::collect()

---

addPriorHistory

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

Description

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

Usage

addPriorHistory(
  x,
  cdm,
  indexDate = "cohort_start_date",
  priorHistoryName = "prior_history",
  tablePrefix = NULL
)
addPriorHistory

Arguments

x          Table with individuals in the cdm
cdm        Object that contains a cdm reference. Use CDMConnector to obtain a cdm reference.
indexDate  Variable in x that contains the date to compute the prior history.
priorHistoryName  name of the new column to be added
tablePrefix  The stem for the permanent tables that will be created. If NULL, temporary tables will be used throughout.

Value

cohort table with added column containing prior history of the individuals

Examples

library(DBI)
library(duckdb)
library(PatientProfiles)
cohort1 <- dplyr::tribble(
  cohort_definition_id = c("1", "1", "1"),
  subject_id = c("1", "2", "3"),
  cohort_start_date = c(
    as.Date("2010-03-03"),
    as.Date("2010-03-01"),
    as.Date("2010-02-01")
  ),
  cohort_end_date = c(
    as.Date("2015-01-01"),
    as.Date("2013-01-01"),
    as.Date("2013-01-01")
  )
)

obs_1 <- dplyr::tribble(
  observation_period_id = c("1", "2", "3"),
  person_id = c("1", "2", "3"),
  observation_period_start_date = c(
    as.Date("2010-02-03"),
    as.Date("2010-02-01"),
    as.Date("2010-01-01")
  ),
  observation_period_end_date = c(
    as.Date("2014-01-01"),
    as.Date("2012-01-01"),
    as.Date("2012-01-01")
  )
)
```r

cdm <-
  mockPatientProfiles(
    seed = 1,
    cohort1 = cohort1,
    observation_period = obs_1
  )

result <- cdm$cohort1 %>% addPriorHistory(cdm)
```

---

### `addSex`

**Compute the sex of the individuals**

**Description**

Compute the sex of the individuals

**Usage**

```r
addSex(x, cdm, sexName = "sex", tablePrefix = NULL)
```

**Arguments**

- **x** Table with individuals in the cdm
- **cdm** Object that contains a cdm reference. Use CDMConnector to obtain a cdm reference.
- **sexName** name of the new column to be added
- **tablePrefix** The stem for the permanent tables that will be created. If NULL, temporary tables will be used throughout.

**Value**

table x with the added column with sex information

**Examples**

```r
library(PatientProfiles)
cdm <- mockPatientProfiles()
cdm$cohort1 %>% addSex(cdm)
```
availableFunctions

Show the available functions for the 4 classifications of data that are supported (numeric, date, binary and categorical)

Usage

availableFunctions(variableType = NULL)

Arguments

variableType A choice between: "numeric", "date", "binary" or "categorical".

Value

A tibble with the available functions for a certain variable classification (or all if NULL)

Examples

library(PatientProfiles)

availableFunctions()

availableFunctions("numeric")

availableFunctions("date")

availableFunctions("binary")

availableFunctions("categorical")

detectVariables

Detect automatically variables with a certain classification

Description

Detect automatically variables with a certain classification

Usage

detectVariables(
  table,
  variableType,
  exclude = c("person_id", "subject_id", "cohort_definition_id", "cohort_name",
             "strata_name", "strata_level")
)
getConceptName

Arguments

- table: Tibble
- variableType: Classification of interest, choice between "numeric", "date", "binary" and "categorical"
- exclude: Variables to exclude

Value

Variables in x with the desired classification

Examples

```r
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)
)
detectVariables(x, "numeric")
```

getConceptName(tableName)

Description

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

Usage

```r
cgetConceptName(tableName)
```

Arguments

- tableName: Name of the table

Value

Name of the concept_id column in that table

Examples

```r
library(PatientProfiles)
ggetConceptName("condition_occurrence")
```
**getEndName**

*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**

`getEndName(tableName)`

**Arguments**

- `tableName`  
  Name of the table

**Value**

Name of the end date column in that table

**Examples**

```r
library(PatientProfiles)
getEndName("condition_occurrence")
```

---

**getStartName**

*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**

`getStartName(tableName)`

**Arguments**

- `tableName`  
  Name of the table

**Value**

Name of the start date column in that table
Examples

```r
library(PatientProfiles)
getStartName("condition_occurrence")
```

Description

It creates a mock database for testing PatientProfiles package

Usage

```r
mockPatientProfiles(
  connectionDetails = list(con = DBI::dbConnect(duckdb::duckdb(), ":memory:"), 
    write_schema = "main", mock_prefix = NULL),
  drug_exposure = NULL,
  drug_strength = NULL,
  observation_period = NULL,
  condition_occurrence = NULL,
  visit_occurrence = NULL,
  concept_ancestor = NULL,
  person = NULL,
  cohort1 = NULL,
  cohort2 = NULL,
  drug_concept_id_size = 5,
  ancestor_concept_id_size = 5,
  condition_concept_id_size = 5,
  visit_concept_id_size = 5,
  ingredient_concept_id_size = 1,
  drug_exposure_size = 10,
  patient_size = 1,
  min_drug_exposure_start_date = "2000-01-01",
  max_drug_exposure_start_date = "2020-01-01",
  earliest_date_of_birth = NULL,
  latest_date_of_birth = NULL,
  earliest_observation_start_date = NULL,
  latest_observation_start_date = NULL,
  min_days_to_observation_end = NULL,
  max_days_to_observation_end = NULL,
  earliest_condition_start_date = NULL,
  latest_condition_start_date = NULL,
  min_days_to_condition_end = NULL,
)
mockPatientProfiles

max_days_to_condition_end = NULL,
earliest_visit_start_date = NULL,
latest_visit_start_date = NULL,
min_days_to_visit_end = NULL,
max_days_to_visit_end = NULL,
seed = 1,
...
)

Arguments

connectionDetails  Connection an details to create the cdm mock object
drug_exposure     default null user can define its own table
drug_strength     default null user can define its own table
observation_period default null user can define its own table
condition_occurrence default null user can define its own table
visit_occurrence default null user can define its own visit_occurrence table
concept_ancestor the concept ancestor table
person                  default null user can define its own table
cohort1                 cohort table for test to run in getindication
cohort2                 cohort table for test to run in getindication
drug_concept_id_size number of unique drug concept id
ancestor_concept_id_size the size of concept ancestor table
condition_concept_id_size number of unique row in the condition concept table
visit_concept_id_size  number of unique visit concept id
visit_occurrence_id_size number of unique visit occurrence id
ingredient_concept_id_size number of unique drug ingredient concept id
drug_exposure_size     number of unique drug exposure
patient_size            number of unique patient
min_drug_exposure_start_date user define minimum drug exposure start date
max_drug_exposure_start_date user define maximum drug exposure start date
earliest\_date\_of\_birth
the earliest date of birth of patient in person table format "dd-mm-yyyy"

latest\_date\_of\_birth
the latest date of birth for patient in person table format "dd-mm-yyyy"

earliest\_observation\_start\_date
the earliest observation start date for patient format "dd-mm-yyyy"

latest\_observation\_start\_date
the latest observation start date for patient format "dd-mm-yyyy"

min\_days\_to\_observation\_end
the minimum number of days of the observational integer

max\_days\_to\_observation\_end
the maximum number of days of the observation period integer

earliest\_condition\_start\_date
the earliest condition start date for patient format "dd-mm-yyyy"

latest\_condition\_start\_date
the latest condition start date for patient format "dd-mm-yyyy"

min\_days\_to\_condition\_end
the minimum number of days of the condition integer

max\_days\_to\_condition\_end
the maximum number of days of the condition integer

earliest\_visit\_start\_date
the earliest visit start date for patient format "dd-mm-yyyy"

latest\_visit\_start\_date
the latest visit start date for patient format "dd-mm-yyyy"

min\_days\_to\_visit\_end
the minimum number of days of the visit integer

max\_days\_to\_visit\_end
the maximum number of days of the visit integer

seed
seed

... user self defined tibble table to put in cdm, it can input as many as the user want

**Value**

cdm of the mock database following user’s specifications

**Examples**

```r
library(PatientProfiles)
cdm <- mockPatientProfiles()
```
This function is used to summarise the dose and/or indication over multiple cohorts.

**Description**

This function is used to summarise the dose and/or indication over multiple cohorts.

**Usage**

```r
summariseCharacteristics(
  cohort,
  cdm,
  strata = list(),
  ageGroup = NULL,
  windowVisitOccurrence = NULL,
  covariates = list(),
  minCellCount = 5
)
```

**Arguments**

- `cohort`: A cohort in the cdm
- `cdm`: A cdm_reference created by CDMConnector
- `strata`: Stratification list
- `ageGroup`: A list of age groups.
- `windowVisitOccurrence`: Window to count visit occurrences.
- `covariates`: Named list of windows to check covariates. The name must point to a cohort-TableName in the cdm.
- `minCellCount`: minimum counts due to obscure

**Value**

A summary of the characteristics of the individuals

**Examples**

```r
library(PatientProfiles)

cdm <- mockPatientProfiles()

summariseCharacteristics(
  cdm$cohort1,
```
summariseResult

Summarise the characteristics of different individuals

**Description**

Summarise the characteristics of different individuals

**Usage**

```r
define ageGroup = list(c(0, 19), c(20, 39), c(40, 59), c(60, 79), c(80, 150)),
windowVisitOccurrence = c(-180, 0)
)

summariseResult(table,
group = list(),
includeOverallGroup = FALSE,
strata = list(),
includeOverallStrata = TRUE,
variables = list(numericVariables = detectVariables(table, "numeric"), dateVariables =
detectVariables(table, "date"), binaryVariables = detectVariables(table, "binary"),
categoricalVariables = detectVariables(table, "categorical")),
functions = list(numericVariables = c("median", "q25", "q75"), dateVariables =
c("median", "q25", "q75"), binaryVariables = c("count", "%"), categoricalVariables =
c("count", "%")),
minCellCount = 5
)
```

**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**
  - List of the different groups of variables, by default they are automatically classified.
- **functions**
  - List of functions to be applied to each one of the group of variables.
- **minCellCount**
  - Minimum count of records to report results.
variableTypes

Value

Table that summarises the characteristics of the individual.

Examples

```r
library(PatientProfiles)
library(dplyr)

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

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

```r
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

addAge, 2
addAttributes, 4
addCategories, 5
addCohortIntersect, 6
addCohortIntersectCount, 9
addCohortIntersectDate, 11
addCohortIntersectDays, 13
addCohortIntersectFlag, 15
addDateOfBirth, 18
addDemographics, 19
addFutureObservation, 20
addInObservation, 22
addIntersect, 23
addPriorHistory, 24
addSex, 26
availableFunctions, 27
detectVariables, 27
getConceptName, 28
getEndName, 29
getStartName, 29
mockPatientProfiles, 30
summariseCharacteristics, 33
summariseResult, 34
variableTypes, 35