Package ‘PatientProfiles’

November 21, 2023

Type Package

Title Identify Characteristics of Patients in the OMOP Common Data Model

Version 0.5.1

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 7.2.3

Suggests covr, duckdb, testthat (>= 3.1.5), knitr, CodelistGenerator, rmarkdown, glue, odbc, ggplot2, spelling, RPostgres, dbplyr, PaRe, here, magick, plotly, ggraph, DT, cowplot, DiagrammeRsvg

Imports magrittr, CDMConnector (>= 1.2.0), dplyr, tidyr, checkmate, lubridate, DBI, rlang, cli, pillar, stringr, gt

VignetteBuilder knitr

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

Language en-US

Depends R (>= 2.10)

Config/testthat/edition 3

Config/testthat/parallel true

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 2023-11-21 14:40:02 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>addAge</td>
<td>3</td>
</tr>
<tr>
<td>addAttributes</td>
<td>4</td>
</tr>
<tr>
<td>addCategories</td>
<td>5</td>
</tr>
<tr>
<td>addCdmName</td>
<td>6</td>
</tr>
<tr>
<td>addCohortIntersect</td>
<td>7</td>
</tr>
<tr>
<td>addCohortIntersectCount</td>
<td>9</td>
</tr>
<tr>
<td>addCohortIntersectDate</td>
<td>12</td>
</tr>
<tr>
<td>addCohortIntersectDays</td>
<td>14</td>
</tr>
<tr>
<td>addCohortIntersectFlag</td>
<td>16</td>
</tr>
<tr>
<td>addCohortName</td>
<td>18</td>
</tr>
<tr>
<td>addConceptIntersect</td>
<td>19</td>
</tr>
<tr>
<td>addConceptIntersectCount</td>
<td>21</td>
</tr>
<tr>
<td>addConceptIntersectDate</td>
<td>22</td>
</tr>
<tr>
<td>addConceptIntersectDays</td>
<td>23</td>
</tr>
<tr>
<td>addConceptIntersectFlag</td>
<td>24</td>
</tr>
<tr>
<td>addDateOfBirth</td>
<td>25</td>
</tr>
<tr>
<td>addDemographics</td>
<td>26</td>
</tr>
<tr>
<td>addFutureObservation</td>
<td>28</td>
</tr>
<tr>
<td>addInObservation</td>
<td>29</td>
</tr>
<tr>
<td>addIntersect</td>
<td>30</td>
</tr>
<tr>
<td>addLargeScaleCharacteristics</td>
<td>32</td>
</tr>
<tr>
<td>addPriorObservation</td>
<td>32</td>
</tr>
<tr>
<td>addSex</td>
<td>34</td>
</tr>
<tr>
<td>availableFunctions</td>
<td>35</td>
</tr>
<tr>
<td>detectVariables</td>
<td>35</td>
</tr>
<tr>
<td>getConceptName</td>
<td>36</td>
</tr>
<tr>
<td>getEndName</td>
<td>37</td>
</tr>
<tr>
<td>getSourceConceptName</td>
<td>37</td>
</tr>
<tr>
<td>getStartName</td>
<td>38</td>
</tr>
<tr>
<td>gtCharacteristics</td>
<td>38</td>
</tr>
<tr>
<td>gtResult</td>
<td>40</td>
</tr>
<tr>
<td>mockPatientProfiles</td>
<td>41</td>
</tr>
<tr>
<td>summariseCharacteristics</td>
<td>44</td>
</tr>
<tr>
<td>summariseLargeScaleCharacteristics</td>
<td>46</td>
</tr>
<tr>
<td>summariseResult</td>
<td>47</td>
</tr>
<tr>
<td>suppressCounts</td>
<td>48</td>
</tr>
<tr>
<td>variableTypes</td>
<td>48</td>
</tr>
</tbody>
</table>

**Index**

50
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 = attr(x, "cdm_reference"),
  indexDate = "cohort_start_date",
  ageName = "age",
  ageGroup = NULL,
  ageDefaultMonth = 1,
  ageDefaultDay = 1,
  ageImposeMonth = FALSE,
  ageImposeDay = FALSE
)

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.

Value

tibble with the age column added
addAttributes

Get attributes from one cohort to another

Examples

```r
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"),
  month_of_birth = c(NA, 07, 08),
  day_of_birth = c(01, 25, 03)
)
cdm <- mockPatientProfiles(person = person, cohort1 = cohort1)
addAge(x = cdm["cohort1"], cdm = cdm)
```

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**  
*Category a numeric variable*

**Description**  
Categorize a numeric variable

**Usage**

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

**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

**Value**
nibble with the categorical variable added.
Examples

```r
#'
library(DBI)
library(duckdb)
library(PatientProfiles)
cohort1 <- dplyr::tibble(
    cohort_definition_id = c("1", "2", "3"),
    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)
        ))
    )
```

**addCdmName**

_Add cdm name_

**Description**

Add cdm name

**Usage**

```
addCdmName(table, cdm = NULL)
```
addCohortIntersect

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()

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 = attr(x, "cdm_reference"),
  targetCohortTable,
  targetCohortId = NULL,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  targetStartDate = "cohort_start_date",
  targetEndDate = "cohort_end_date",
  window = list(c(0, Inf)),
  order = "first",
  flag = TRUE,
  count = TRUE,
  date = TRUE,
  days = TRUE,
  nameStyle = "{value}_{cohort_name}_{window_name}"
)
addCohortIntersect

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.
- **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
- **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

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"
  ),
)
)
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

```r
addCohortIntersectCount(
  x,
  cdm = attr(x, "cdm_reference"),
  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}"
)
```

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.
- `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

Value

table with added columns with overlap information

Examples

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

cohort1 <- dplyr::tibble(
  cohort_definition_id = c(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"
    )
  )
)
)
}

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

result <- cdm$cohort1 %>%
  addCohortIntersectCount(
    targetCohortTable = "cohort2"
  )
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 = attr(x, "cdm_reference"),
  targetCohortTable,
  targetCohortId = NULL,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  targetDate = "cohort_start_date",
  order = "first",
  window = c(0, Inf),
  nameStyle = "{cohort_name}_{window_name}"
)

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.
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
Value

x along with additional columns for each cohort of interest.

Examples

```r
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",
    "2020-01-29",
    "2020-01-29"))
)```
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 = attr(x, "cdm_reference"), 
  targetCohortTable, 
  targetCohortId = NULL, 
  indexDate = "cohort_start_date", 
  censorDate = NULL, 
  targetDate = "cohort_start_date", 
  order = "first", 
  window = c(0, Inf), 
  nameStyle = "{cohort_name}_{window_name}" 
)

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
**addCohortIntersectDays**

- **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.

**Value**

x along with additional columns for each cohort of interest.

**Examples**

```r
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"
  )
)

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),
  subject_id = c(1, 1, 1, 2, 2),
  cohort_start_date = as.Date(c(
    "2020-01-15",
    "2020-01-20",
    "2020-01-01",
    "2020-02-01"
  )
)
)
```
addCohortIntersectFlag

It creates columns to indicate the presence of cohorts

Description
It creates columns to indicate the presence of cohorts

Usage
addCohortIntersectFlag(
  x,
  cdm = attr(x, "cdm_reference"),
  targetCohortTable,
  targetCohortId = NULL,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  targetStartDate = "cohort_start_date",
  targetEndDate = "cohort_end_date"
)
addCohortIntersectFlag

targetEndDate = "cohort_end_date",
window = list(c(0, Inf)),
nameStyle = "{cohort_name}\_\{window_name\}"
)

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.
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

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")
  )
)
addCohortName <- dplyr::tibble(
  cohort_definition_id = c(1, 1, 1, 1, 1, 1, 1),
  subject_id = c(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"))
)

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

---

**addCohortName**

*Add cohort name for each cohort_definition_id*

**Description**

Add cohort name for each cohort_definition_id

**Usage**

`addCohortName(cohort)`
addConceptIntersect

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()

Description

It creates columns to indicate overlap information between a table and a concept

Usage

addConceptIntersect(
  x,
  conceptSet,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetStartDate = "cohort_start_date",
  targetEndDate = NULL,
  order = "first",
  flag = TRUE,
  count = TRUE,
  date = TRUE,
  days = TRUE,
  nameStyle = "\{value\}_{concept_name}_{window_name}")

### 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**
  - 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**
  - Which record is considered in case of multiple records

- **flag**
  - TRUE or FALSE. If TRUE, flag will be 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

### Value

Table with added columns with overlap information

### Examples

```r
library(PatientProfiles)
library(CodelistGenerator)

cdm <- mockPatientProfiles()
library(PatientProfiles)
library(CodelistGenerator)

# result <- cdm$cohort1 %>%
#   addConceptIntersect(
#     conceptSet = getDrugIngredientCodes(cdm, "acetaminophen")
#   ) %>%
#   dplyr::collect()
```
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 = "cohort_start_date",
  targetEndDate = NULL,
  order = "first",
  nameStyle = "{concept_name}_{window_name}"
)

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 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

Value
table with added columns with overlap information
addConceptIntersectDate

Examples

```r
library(PatientProfiles)
library(CodelistGenerator)

cdm <- mockPatientProfiles()
# result <- cdm$cohort1 %>%
#  addConceptIntersectCount(
#    conceptSet = getDrugIngredientCodes(cdm, "acetaminophen")
#  ) %>%
#  dplyr::collect()
```

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

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

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` date of reference in cohort table
- `order` last or first date to use for date/time calculations.
- `nameStyle` naming of the added column or columns, should include required parameters
addConceptIntersectDays

Value

table with added columns with overlap information

Examples

library(PatientProfiles)
library(CodelistGenerator)

cdm <- mockPatientProfiles()
# result <- cdm$cohort1 %>%
#   addConceptIntersectDate(
#     conceptSet = getDrugIngredientCodes(cdm, "acetaminophen")
#   ) %>%
#   dplyr::collect()

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 = "cohort_start_date",
  order = "first",
  nameStyle = "{concept_name}_{window_name}"
)

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.
addConceptIntersectFlag

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 = "cohort_start_date",
  targetEndDate = NULL,
  order = "first",
  nameStyle = "{concept_name}_{window_name}"
)
Arguments

x Table with individuals in the cdm
cconceptSet 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 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.
namesStyle naming of the added column or columns, should include required parameters

Value
table with added columns with overlap information

Examples

library(PatientProfiles)
library(CodelistGenerator)

cdm <- mockPatientProfiles()
# result <- cdm$cohort1 %>%
#   addConceptIntersectFlag(
#     conceptSet = getDrugIngredientCodes(cdm, "acetaminophen")
#   ) %>%
#   dplyr::collect()

addDateOfBirth

Add a column with the individual birth date

Description

Add a column with the individual birth date

Usage

addDateOfBirth(
  x,
  cdm = attr(x, "cdm_reference"),
  name = "date_of_birth",
  missingDay = 1,
)
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

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

Description

Compute demographic characteristics at a certain date

Usage

```r
addDemographics(
  x,
  cdm = attr(x, "cdm_reference"),
  indexDate = "cohort_start_date",
  age = TRUE,
  ageName = "age",
  ageDefaultMonth = 1,
  ageDefaultDay = 1,
  ageImposeMonth = FALSE,
)```
addDemographics

ageImposeDay = FALSE,
ageGroup = NULL,
sex = TRUE,
sexName = "sex",
priorObservation = TRUE,
priorObservationName = "prior_observation",
futureObservation = TRUE,
futureObservationName = "future_observation"
)

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
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
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
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

Value

cohort table with the added demographic information columns
**Examples**

```r
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

**Usage**

```r
addFutureObservation(
  x,
  cdm = attr(x, "cdm_reference"),
  indexDate = "cohort_start_date",
  futureObservationName = "future_observation"
)
```

**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

**Value**

cohort table with added column containing future observation of the individuals

**Examples**

```r
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(
    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")
  )
)

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

result <- cdm$cohort1 %>% addFutureObservation(cdm)

---

addInObservation  
Indicate if a certain record is within the observation period

**Description**

Indicate if a certain record is within the observation period

**Usage**

```r
addInObservation(
  x,
  cdm = attr(x, "cdm_reference"),
  indexDate = "cohort_start_date",
```
name = "in_observation"
)

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

Value

cohort table with the added binary column assessing inObservation

Examples

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

Usage

addIntersect(
  x,
  cdm = attr(x, "cdm_reference"),
  tableName,
  value,
  filterVariable = NULL,
  filterId = NULL,
  idName = NULL,
  window = list(c(0, Inf)),
  indexDate = "cohort_start_date",
  censorDate = NULL,
  targetStartDate = getStartName(tableName),
  targetEndDate = getEndName(tableName),
)
```r
addIntersect

order = "first",
nameStyle = "\{value\}_{id_name}_{window_name}"
)

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)
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.
censorDate whether to censor overlap events at a date column 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)
order last or first date to use for date/time calculations
nameStyle naming of the added column or columns, should include required parameters

Value

table with added columns with overlap information

Examples

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

*This function is used to add columns with the large scale characteristics of a cohort table.*

**Description**

This function is used to add columns with the large scale characteristics of a cohort table.

**Usage**

```r
addLargeScaleCharacteristics(
  cohort,
  window = list(c(0, Inf)),
  eventInWindow = NULL,
  episodeInWindow = NULL,
  includeSource = FALSE,
  minimumFrequency = 0.005
)
```

**Arguments**

- `cohort`: The cohort to characterise.
- `window`: Temporal windows that we want to characterize.
- `eventInWindow`: Tables to characterise the events in the window.
- `episodeInWindow`: Tables to characterise the episodes in the window.
- `includeSource`: Whether to include source concepts.
- `minimumFrequency`: Minimum frequency covariates to report.

**Value**

The output of this function is the cohort with the new created columns

---

**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
addPriorObservation

Usage

addPriorObservation(
  x,
  cdm = attr(x, "cdm_reference"),
  indexDate = "cohort_start_date",
  priorObservationName = "prior_observation"
)

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 observation.
  priorObservationName  name of the new column to be added

Value

  cohort table with added column containing prior 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(
    as.Date("2010-02-03"),
    as.Date("2010-02-01"),
    as.Date("2010-01-01")
  ),
addSex

observation_period_end_date = c(
  as.Date("2014-01-01"),
  as.Date("2012-01-01"),
  as.Date("2012-01-01")
)

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

result <- cdm$cohort1 %>% addPriorObservation(cdm)

---

addSex  Compute the sex of the individuals

Description
Compute the sex of the individuals

Usage
addSex(x, cdm = attr(x, "cdm_reference"), sexName = "sex")

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

Value

table x with the added column with sex information

Examples

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)*

#### Description

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

#### Usage

```r
availableFunctions(variableType = NULL)
```

#### Arguments

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>variableType</td>
<td>A choice between: &quot;numeric&quot;, &quot;date&quot;, &quot;binary&quot; or &quot;categorical&quot;.</td>
</tr>
</tbody>
</table>

#### Value

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

#### Examples

```r
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

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

<table>
<thead>
<tr>
<th>table</th>
<th>Tibble</th>
</tr>
</thead>
<tbody>
<tr>
<td>variableType</td>
<td>Classification of interest, choice between &quot;numeric&quot;, &quot;date&quot;, &quot;binary&quot; and &quot;categorical&quot;</td>
</tr>
<tr>
<td>exclude</td>
<td>Variables to exclude</td>
</tr>
</tbody>
</table>

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

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

Description

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

Usage

getConceptName(tableName)

Arguments

| tableName     | Name of the table |

Value

Name of the concept_id column in that table

Examples

```r
library(PatientProfiles)
getConceptName("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**

```r
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")
```

---

**getSourceConceptName**

*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**

```r
ggetSourceConceptName(tableName)
```

**Arguments**
- `tableName` Name of the table

**Value**
Name of the source_concept_id column in that table
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

library(PatientProfiles)
getStartName("condition_occurrence")

gtCharacteristics

Create a gt table from a summarisedCharacteristics object.

Description

‘r lifecycle::badge("experimental")’
gtCharacteristics

Usage

gtCharacteristics(
    summarisedCharacteristics,
    pivotWide = c("CDM Name", "Group", "Strata"),
    format = c("N (%)" = "count (percentage%)", "median [min; q25 - q75; max]",
               "mean (sd)", "median [q25 - q75]", N = "count"),
    keepNotFormatted = TRUE,
    decimals = c(default = 0),
    decimalMark = ".",
    bigMark = "",
)

Arguments

summarisedCharacteristics
    Summary characteristics long table.

pivotWide
    variables to pivot wide

format
    formats and labels to use

keepNotFormatted
    Wheather to keep not formated estimate types

decimals
    Decimals per estimate_type

decimalMark
    decimal mark

bigMark
    big mark

Value

New table in gt format

Examples

library(PatientProfiles)

cdm <- mockPatientProfiles()

summariseCharacteristics(
    cohort = cdm$cohort1,
    ageGroup = list(c(0, 19), c(20, 39), c(40, 59), c(60, 79), c(80, 150)),
    tableIntersect = list(
        "Visits" = list(
            tableName = "visit_occurrence", value = "count", window = c(-365, 0)
        ),
    ),
    cohortIntersect = list(
        "Medications" = list(
            targetCohortTable = "cohort2", value = "flag", window = c(-365, 0)
        ),
    ),
    minCellCount = 1


\begin{verbatim}
  ) %>%
  gtCharacteristics()

---

\textbf{gtResult} \hspace{1cm} \textit{Create a gt table from a summary object.}

\textbf{Description}

'\texttt{r lifecycle::badge("experimental")}'

\textbf{Usage}

\begin{verbatim}
  gtResult(
    summarisedResult,
    long,
    wide,
    format = c("N (%)" = "count (percentage%)", "median [min; q25 - q75; max]",
              "mean (sd)", "median [q25 - q75]", N = "count"),
    keepNotFormatted = TRUE,
    decimals = c(default = 0),
    decimalMark = ",",
    bigMark = ","
  )
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{summarisedResult} \hspace{1cm} A SummarisedResult object.
  \item \texttt{long} \hspace{1cm} List of variables and specification to long
  \item \texttt{wide} \hspace{1cm} List of variables and specification to wide
  \item \texttt{format} \hspace{1cm} formats and labels to use
  \item \texttt{keepNotFormatted} \hspace{1cm} Whether to keep not formatted estimate types
  \item \texttt{decimals} \hspace{1cm} Decimals per estimate_type
  \item \texttt{decimalMark} \hspace{1cm} decimal mark
  \item \texttt{bigMark} \hspace{1cm} big mark
\end{itemize}

\textbf{Value}

A formatted summarisedResult gt object.
\end{verbatim}
Examples

```r
library(PatientProfiles)

cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  summariseCharacteristics(
    ageGroup = list(c(0, 19), c(20, 39), c(40, 59), c(60, 79), c(80, 150)),
    minCellCount = 1
  ) %>%
  gtResult(
    long = list(
      "Variable" = c(level = "variable", "clean"),
      "Level" = c(level = "variable_level"),
      "Format" = c(level = "format", "separator-right")
    ),
    wide = list(
      "CDM Name" = c(level = "cdm_name"),
      "Group" = c(level = c("group_name", "group_level")),
      "Strata" = c(level = c("strata_name", "strata_level"))
    ),
    format = c(
      "N (%)" = "count (percentage%)",
      "N" = "count",
      "median [Q25-Q75]" = "median [q25-q75]"
    ),
    decimals = c(count = 0),
    keepNotFormatted = FALSE
  )
```

---

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

**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,
visit_occurrence_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,
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 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"
summariseCharacteristics

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

... 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

library(PatientProfiles)
cdm <- mockPatientProfiles()

summariseCharacteristics

Summarise characteristics of individuals

Description

Summarise characteristics of individuals

Usage

summariseCharacteristics(
  cohort,
  cdm = attr(cohort, "cdm_reference"),
  strata = list(),
  demographics = TRUE,
  ageGroup = NULL,
  tableIntersect = list(),
  cohortIntersect = list(),
  conceptIntersect = list(),
  otherVariables = character(),
  minCellCount = 5
)
**Arguments**

- **cohort**: A cohort in the cdm
- **cdm**: A cdm reference.
- **strata**: Stratification list
- **demographics**: Whether to summarise demographics data.
- **ageGroup**: A list of age groups.
- **tableIntersect**: A list of arguments that uses addTableIntersect function to add variables to summarise
- **cohortIntersect**: A list of arguments that uses addCohortIntersect function to add variables to summarise.
- **conceptIntersect**: A list of arguments that uses addConceptIntersect function to add variables to summarise.
- **otherVariables**: Other variables contained in cohort that you want to be summarised.
- **minCellCount**: minimum counts due to obscure

**Value**

A summary of the characteristics of the individuals

**Examples**

```r
library(PatientProfiles)

cdm <- mockPatientProfiles()

summariseCharacteristics(
  cohort = cdm$cohort1,
  ageGroup = list(c(0, 19), c(20, 39), c(40, 59), c(60, 79), c(80, 150)),
  tableIntersect = list(
    "Visits" = list(
      tableName = "visit_occurrence", value = "count", window = c(-365, 0)
    ),
  ),
  cohortIntersect = list(
    "Medications" = list(
      targetCohortTable = "cohort2", value = "flag", window = c(-365, 0)
    ),
  )
)
```
summariseLargeScaleCharacteristics

This function is used to summarise the large scale characteristics of a cohort table

Description

This function is used to summarise the large scale characteristics of a cohort table

Usage

```r
summariseLargeScaleCharacteristics(
  cohort,
  strata = list(),
  window = list(c(-Inf, -366), c(-365, -31), c(-30, -1), c(0, 0), c(1, 30), c(31, 365),
                c(366, Inf)),
  eventInWindow = NULL,
  episodeInWindow = NULL,
  includeSource = FALSE,
  minCellCount = 5,
  minimumFrequency = 0.005,
  cdm = attr(cohort, "cdm_reference")
)
```

Arguments

- `cohort`: The cohort to characterise.
- `strata`: Stratification list.
- `window`: Temporal windows that we want to characterize.
- `eventInWindow`: Tables to characterise the events in the window.
- `episodeInWindow`: Tables to characterise the episodes in the window.
- `includeSource`: Whether to include source concepts.
- `minCellCount`: All counts lower than minCellCount will be obscured.
- `minimumFrequency`: Minimum frequency covariates to report.
- `cdm`: A cdm reference.

Value

The output of this function is a ‘ResultSummary' containing the relevant information.
summariseResult

```r
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", "min", "q25", "q75", "max"),
                   dateVariables = c("median", "min", "q25", "q75", "max"),
                   binaryVariables = c("count", "percentage"),
                   categoricalVariables = c("count", "percentage")),
  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.

**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)
```

### suppressCounts

**Function to suppress counts in summarised objects**

**Description**

Function to suppress counts in summarised objects

**Usage**

```r
suppressCounts(result, minCellCount = 5)
```

**Arguments**

- `result`: SummarisedResult object
- `minCellCount`: Minimum count of records to report results.

**Value**

Table with suppressed counts

### 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**

```r
variableTypes(table)
```
variableTypes

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

addAge, 3
addAttributes, 4
addCategories, 5
addCdmName, 6
addCohortIntersect, 7
addCohortIntersectCount, 9
addCohortIntersectDate, 12
addCohortIntersectDays, 14
addCohortIntersectFlag, 16
addCohortName, 18
addConceptIntersect, 19
addConceptIntersectCount, 21
addConceptIntersectDate, 22
addConceptIntersectDays, 23
addConceptIntersectFlag, 24
addDateOfBirth, 25
addDemographics, 26
addFutureObservation, 28
addInObservation, 29
addIntersect, 30
addLargeScaleCharacteristics, 32
addPriorObservation, 32
addSex, 34
availableFunctions, 35
detectVariables, 35
getConceptName, 36
getEndName, 37
ggetSourceConceptName, 37
getStartName, 38
gtCharacteristics, 38
gtResult, 40
mockPatientProfiles, 41
summariseCharacteristics, 44
summariseLargeScaleCharacteristics, 46
summariseResult, 47
suppressCounts, 48

variableTypes, 48

50