Package ‘PatientProfiles’

April 11, 2024

Type Package

Title Identify Characteristics of Patients in the OMOP Common Data Model

Version 0.8.0

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

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

License Apache License (>= 2)

Encoding UTF-8

RoxygenNote 7.3.1

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

Imports magrittr, CDMConnector (>= 1.3.1), dplyr, tidyr, checkmate, lubridate, rlang, cli, stringr, omopgenerics (>= 0.1.2), visOmopResults (>= 0.2.0), lifecycle, purrr

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

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

Language en-US

Depends R (>= 2.10)

Config/testthat/edition 3

Config/testthat/parallel true

VignetteBuilder knitr

NeedsCompilation no

Author Marti Catala [aut, cre] (<https://orcid.org/0000-0003-3308-9905>), Yuchen Guo [aut] (<https://orcid.org/0000-0002-0847-4855>), Mike Du [aut] (<https://orcid.org/0000-0002-9517-8834>), Kim Lopez-Guell [aut] (<https://orcid.org/0000-0002-8462-8688>),
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 2024-04-11 14:00:02 UTC

R topics documented:

addAge . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
addCategories . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
addCdmName . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
addCohortIntersect . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
addCohortIntersectCount . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
addCohortIntersectDate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
addCohortIntersectDays . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
addCohortIntersectFlag . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
addCohortName . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
addConceptIntersect . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
addConceptIntersectCount . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
addConceptIntersectDate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
addConceptIntersectDays . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
addConceptIntersectFlag . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
addDateOfBirth . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
addDeathDate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
addDeathDays . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23
addDeathFlag . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
addDemographics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
addFutureObservation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
addInObservation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
addIntersect . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
addLargeScaleCharacteristics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 30
addPriorObservation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
addSex . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
addTableIntersect . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
addTableIntersectCount . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
addTableIntersectDate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
addTableIntersectDays . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
addTableIntersectField . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
addTableIntersectFlag . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
assertNameStyle . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
availableEstimates . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 41
availableFunctions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 42
endDateColumn . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
formatCharacteristics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
gtCharacteristics . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 44
gtResult . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45
mockPatientProfiles . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
addAge

Description
Compute the age of the individuals at a certain date

Usage
addAge(
x,  
cdm = lifecycle::deprecated(),  
indexDate = "cohort_start_date",  
ageName = "age",  
)
addAge

ageGroup = NULL,
ageDefaultMonth = 1,
ageDefaultDay = 1,
ageImposeMonth = FALSE,
ageImposeDay = FALSE,
missingAgeGroupValue = "None"
)

Arguments

x Table with individuals in the cdm.
cdm A cdm_reference object.
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.
missingAgeGroupValue Value to include if missing age.

Value
tibble with the age column added.

Examples

cdm <- mockPatientProfiles()
cdm$cohort1 |> addAge()
CDMConnector::cdmDisconnect(cdm = cdm)
addCategories

Categorize a numeric variable

Description
Categorize a numeric variable

Usage
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
tibble with the categorical variable added.

Examples

cdm <- mockPatientProfiles()
result <- cdm$cohort1 %>%
  addAge() %>%
  addCategories(
    variable = "age",
    categories = list("age_group" = list(
      "0 to 39" = c(0, 39), 
      "40 to 79" = c(40, 79), 
      "80 to 150" = c(80, 150)
    ))
  )
CDMConnector::cdmDisconnect(cdm = cdm)
## addCdmName

### Description

Add cdm name

### Usage

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

### Arguments

- `table`: Table in the cdm
- `cdm`: A cdm reference object

### Value

Table with an extra column with the cdm names

### Examples

```r
library(PatientProfiles)

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

## addCohortIntersect

### 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

```r
addCohortIntersect(cdm1, cdm2, table1, table2, num_occ = FALSE, presence = FALSE, date = FALSE, time_diff = FALSE)
```

### Arguments

- `cdm1`: A cdm reference object
- `cdm2`: A cdm reference object
- `table1`: The table from `cdm1`
- `table2`: The table from `cdm2`
- `num_occ`: A logical value indicating whether to compute the number of occurrences
- `presence`: A logical value indicating whether to compute a flag of presence
- `date`: A logical value indicating whether to compute a date
- `time_diff`: A logical value indicating whether to compute a time difference

### Value

Table with the intersect of two cohorts

### Examples

```r
library(PatientProfiles)

cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addCohortIntersect(dgm2, num_occ = TRUE, presence = TRUE, date = TRUE, time_diff = TRUE)
```
addCohortIntersect

Usage

addCohortIntersect(
  x,
  cdm = lifecycle::deprecated(),
  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}")

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.
addCohortIntersectCount

Examples

```r
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addCohortIntersect(
    targetCohortTable = "cohort2"
  )
CDMConnector::cdmDisconnect(cdm = cdm)
```

Description

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

Usage

```r
addCohortIntersectCount(
  x,
  cdm = lifecycle::deprecated(),
  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.
addCohortIntersectDate

**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
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addCohortIntersectCount(
    targetCohortTable = "cohort2"
  )
CDMConnector::cdmDisconnect(cdm = cdm)
```

---

**addCohortIntersectDate**

*Date of cohorts that are present in a certain window*

**Description**

Date of cohorts that are present in a certain window

**Usage**

```r
addCohortIntersectDate(
  x,
  cdm = lifecycle::deprecated(),
  targetCohortTable,
  targetCohortId = NULL,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  targetDate = "cohort_start_date",
  order = "first",
  window = c(0, Inf),
  nameStyle = "{cohort_name}_{window_name}"
)
```
**addCohortIntersectDays**

**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
cdm <- mockPatientProfiles()

# Example

addCohortIntersectDate(
  targetCohortTable = "cohort2"
)

CDMConnector::cdmDisconnect(cdm = cdm)
```

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

Usage

```r
addCohortIntersectDays(
  x,
  cdm = lifecycle::deprecated(),
  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 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
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addCohortIntersectDays(
    targetCohortTable = "cohort2"
  )
CDMConnector::cdmDisconnect(cdm = cdm)
```
addCohortIntersectFlag

It creates columns to indicate the presence of cohorts

Description

It creates columns to indicate the presence of cohorts

Usage

addCohortIntersectFlag(
  x,
  cdm = lifecycle::deprecated(),
  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.
namStyle naming of the added column or columns, should include required parameters.

Value

table with added columns with overlap information.
addCohortName

Examples

```r
 library(PatientProfiles)

cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addCohortIntersectFlag(
    targetCohortTable = "cohort2"
  )
CDMConnector::cdmDisconnect(cdm = cdm)
```

---

### addCohortName

*Add cohort name for each cohort_definition_id*

### Description

Add cohort name for each cohort_definition_id

### Usage

```r
 addCohortName(cohort)
```

### Arguments

- `cohort` cohort to which add the cohort name

### Value

cohort with an extra column with the cohort names

### Examples

```r
 library(PatientProfiles)

cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addCohortName()
```
addConceptIntersect  

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

**Description**

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

**Usage**

```r
addConceptIntersect(
  x,
  conceptSet,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetStartDate = "event_start_date",
  targetEndDate = "event_end_date",
  order = "first",
  value = c("flag", "count", "date", "days"),
  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**: Event start date to use for the intersection.
- **targetEndDate**: Event end date to use for the intersection.
- **order**: last or first date to use for date/days calculations.
- **value**: Choices between c("value", "flag", "days", "date").
- **nameStyle**: naming of the added column or columns, should include required parameters.

**Value**

```
table with added columns with overlap information
```
Examples

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

**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**

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

**Arguments**

- **x** Table with individuals in the cdm.
- **conceptSet** Concept set list.
- **indexDate** Variable in x that contains the date to compute the intersection.
- **censorDate** whether to censor overlap events at a date column of x
- **window** window to consider events in.
- **targetStartDate** Event start date to use for the intersection.
- **targetEndDate** Event end date to use for the intersection.
- **nameStyle** naming of the added column or columns, should include required parameters.

**Value**

table with added columns with overlap information

**Examples**

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

*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 = "event_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**: Event date to use for the intersection.
- **order**: last or first date to use for date/days calculations.
- **nameStyle**: naming of the added column or columns, should include required parameters.

*Value*

table with added columns with overlap information

*Examples*

```r
library(PatientProfiles)
cdm <- mockPatientProfiles()
concept <- dplyr::tibble(
  concept_id = c(1125315),
  domain_id = "Drug",
  vocabulary_id = NA_character_,
```
addConceptIntersectDays

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

Description

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

Usage

addConceptIntersectDays(
  x,
  conceptSet,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetDate = "event_start_date",
  order = "first",
  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**

- **targetDate**: Event date to use for the intersection.
- **order**: last or first date to use for date/days calculations.
- **nameStyle**: naming of the added column or columns, should include required parameters.

**Value**

table with added columns with overlap information

**Examples**

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

---

**addConceptIntersectFlag**

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

**Description**

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

Usage

addConceptIntersectFlag(  
  x,  
  conceptSet,  
  indexDate = "cohort_start_date",  
  censorDate = NULL,  
  window = list(c(0, Inf)),  
  targetStartDate = "event_start_date",  
  targetEndDate = "event_end_date",  
  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  Event start date to use for the intersection.
targetEndDate  Event end date to use for the intersection.
namensStyle  naming of the added column or columns, should include required parameters.

Value

table with added columns with overlap information

Examples

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

conceptSet = list("acetaminophen"=1125315)
) %>%
dplyr::collect()
CDMConnector::cdmDisconnect(cdm = cdm)

addDateOfBirth

Add a column with the individual birth date

Description

Add a column with the individual birth date

Usage

addDateOfBirth(
  x,
  cdm = lifecycle::deprecated(),
  name = "date_of_birth",
  missingDay = 1,
  missingMonth = 1,
  imposeDay = FALSE,
  imposeMonth = FALSE
)

Arguments

<table>
<thead>
<tr>
<th>x</th>
<th>Table in the cdm that contains 'person_id' or 'subject_id'.</th>
</tr>
</thead>
<tbody>
<tr>
<td>cdm</td>
<td>A cdm_reference object.</td>
</tr>
<tr>
<td>name</td>
<td>Name of the column to be added with the date of birth.</td>
</tr>
<tr>
<td>missingDay</td>
<td>Day of the individuals with no or imposed day of birth.</td>
</tr>
<tr>
<td>missingMonth</td>
<td>Month of the individuals with no or imposed month of birth.</td>
</tr>
<tr>
<td>imposeDay</td>
<td>Whether to impose day of birth.</td>
</tr>
<tr>
<td>imposeMonth</td>
<td>Whether to impose month of birth.</td>
</tr>
</tbody>
</table>

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()
CDMConnector::cdmDisconnect(cdm = cdm)
addDeathDate

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

**Description**

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

**Usage**

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

**Arguments**

- `x` Table with individuals in the cdm.
- `indexDate` Variable in x that contains the window origin.
- `censorDate` Name of a column to stop followup.
- `window` window to consider events over.
- `deathDateName` name of the new column to be added.

**Value**

table x with the added column with death information added.

**Examples**

```r
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addDeathDate()
CDMConnector::cdmDisconnect(cdm = cdm)
```
addDeathDays

Add days to death for individuals. Only death within the same observation period than ‘indeDate’ will be observed.

Description

Add days to death for individuals. Only death within the same observation period than ‘indeDate’ will be observed.

Usage

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

Arguments

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

Value

table x with the added column with death information added.

Examples

cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addDeathDays()
CDMConnector::cdmDisconnect(cdm = cdm)
addDeathFlag

Add flag for death for individuals. Only death within the same observation period than ‘indeDate’ will be observed.

Description

Add flag for death for individuals. Only death within the same observation period than ‘indeDate’ will be observed.

Usage

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

Arguments

x Table with individuals in the cdm.
indexDate Variable in x that contains the window origin.
censorDate Name of a column to stop followup.
window window to consider events over.
deadthFlagName name of the new column to be added.

Value

table x with the added column with death information added.

Examples

```r
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addDeathFlag()
CDMConnector::cdmDisconnect(cdm = cdm)
```
addDemographics

Compute demographic characteristics at a certain date

Description
Compute demographic characteristics at a certain date

Usage
addDemographics(
  x,
  cdm = lifecycle::deprecated(),
  indexDate = "cohort_start_date",
  age = TRUE,
  ageName = "age",
  ageDefaultMonth = 1,
  ageDefaultDay = 1,
  ageImposeMonth = FALSE,
  ageImposeDay = FALSE,
  ageGroup = NULL,
  missingAgeGroupValue = "None",
  sex = TRUE,
  sexName = "sex",
  missingSexValue = "None",
  priorObservation = TRUE,
  priorObservationName = "prior_observation",
  priorObservationType = "days",
  futureObservation = TRUE,
  futureObservationName = "future_observation",
  futureObservationType = "days",
  dateOfBirth = FALSE,
  dateOfBirthName = "date_of_birth"
)

Arguments

x Table with individuals in the cdm.
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.
addDemographics

ageImposeMonth  TRUE or FALSE. Whether the month of the date of birth will be considered as missing for all the individuals.

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

ageGroup        if not NULL, a list of ageGroup vectors.

missingAgeGroupValue  Value to include if missing age.

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

sexName         Sex variable name.

missingSexValue  Value to include if missing sex.

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

priorObservationName  Prior observation variable name.

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

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

futureObservationName  Future observation variable name.

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

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

dateOfBirthName  dateOfBirth column name.

Value

cohort table with the added demographic information columns.

Examples

library(PatientProfiles)
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
   addDemographics()
CDMConnector::cdmDisconnect(cdm = cdm)
addFutureObservation

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

Description

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

Usage

```
addFutureObservation(
  x,
  cdm = lifecycle::deprecated(),
  indexDate = "cohort_start_date",
  futureObservationName = "future_observation",
  futureObservationType = "days"
)
```

Arguments

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

Value

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

Examples

```
cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addFutureObservation()
CDMConnector::cdmDisconnect(cdm = cdm)
```
addInObservation

Indicate if a certain record is within the observation period

Description

Indicate if a certain record is within the observation period

Usage

```r
addInObservation(
  x,
  cdm = lifecycle::deprecated(),
  indexDate = "cohort_start_date",
  window = c(0, 0),
  completeInterval = FALSE,
  name = lifecycle::deprecated(),
  nameStyle = "in_observation"
)
```

Arguments

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

Value

cohort table with the added binary column assessing inObservation.

Examples

```r
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addInObservation()
CDMConnector::cdmDisconnect(cdm = cdm)
```
**addIntersect**

*It creates columns to indicate overlap information between two tables*

**Description**

`r lifecycle::badge("deprecated")`

**Usage**

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

**Arguments**

- **x** Table with individuals in the cdm.
- **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.
addLargeScaleCharacteristics

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

Description

`r lifecycle::badge("experimental")`

Usage

```r
addLargeScaleCharacteristics(
  cohort,
  window = list(c(0, Inf)),
  eventInWindow = NULL,
  episodeInWindow = NULL,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  minimumFrequency = 0.005,
  excludedCodes = NULL
)
```

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.
- **indexDate**: Variable in `x` that contains the date to compute the intersection.
addPriorObservation

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

Description

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

Usage

addPriorObservation(
  x, 
  cdm = lifecycle::deprecated(), 
  indexDate = "cohort_start_date", 
  priorObservationName = "prior_observation", 
  priorObservationType = "days"
)

Arguments

x Table with individuals in the cdm.

cdm A cdm_reference object.

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

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

Examples

cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addPriorObservation()
CDMConnector::cdmDisconnect(cdm = cdm)

addSex
  Compute the sex of the individuals

Description
  Compute the sex of the individuals

Usage

  addSex(
    x,
    cdm = lifecycle::deprecated(),
    sexName = "sex",
    missingSexValue = "None"
  )

Arguments

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

Value
  table x with the added column with sex information.
Examples

cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  addSex()
CDMConnector::cdmDisconnect(cdm = cdm)

addTableIntersect  Compute the intersect with an omop table, you can compute the number of occurrences, a flag of presence, a certain date, the time difference and/or obtain a certain column.

Description

Compute the intersect with an omop table, you can compute the number of occurrences, a flag of presence, a certain date, the time difference and/or obtain a certain column.

Usage

addTableIntersect(
  x,
  tableName,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  order = "first",
  targetStartDate = startDateColumn(tableName),
  targetEndDate = endDateColumn(tableName),
  flag = TRUE,
  count = TRUE,
  date = TRUE,
  days = TRUE,
  field = character(),
  nameStyle = "{table_name}_{value}_{window_name}"
)

Arguments

x  Table with individuals in the cdm.

tableName  Name of the table to intersect with. Options: visit_occurrence, condition_occurrence, drug_exposure, procedure_occurrence, device_exposure, measurement, observation, drug_era, condition_era, specimen.

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.
addTableIntersectCount

Compute number of intersect with an omop table.

Description

Compute number of intersect with an omop table.

Usage

addTableIntersectCount(
  x,
  tableName,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window to consider events in.
  order which record is considered in case of multiple records (only required for date
  and days options).
  targetStartDate Column name with start date for comparison.
  targetEndDate Column name with end date for comparison.
  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.
  field Other columns from the table to intersect.
  nameStyle naming of the added column or columns, should include required parameters.

Value

table with added columns with intersect information.

Examples

cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addTableIntersect(tableName = "visit_occurrence")
CMDConnector::cdmDisconnect(cdm = cdm)
addTableIntersectDate

window = list(c(0, Inf)),
targetStartDate = startDateColumn(tableName),
targetEndDate = endDateColumn(tableName),
nameStyle = "{table_name}_{(window_name)}"
)

Arguments

x Table with individuals in the cdm.
tableName Name of the table to intersect with. Options: visit_occurrence, condition_occurrence,
drug_exposure, procedure_occurrence, device_exposure, measurement, observation, drug_era, condition_era, specimen.
indexDate Variable in x that contains the date to compute the intersection.
censorDate whether to censor overlap events at a specific date or a column date of x.
window window to consider events in.
targetStartDate Column name with start date for comparison.
targetEndDate Column name with end date for comparison.
nameStyle naming of the added column or columns, should include required parameters.

Value
table with added columns with intersect information.

Examples

cdm <- mockPatientProfiles()

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

CDMConnector::cdmDisconnect(cdm = cdm)

addTableIntersectDate  Compute date of intersect with an omop table.

Description
Compute date of intersect with an omop table.
Usage

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

Arguments

- **x**: Table with individuals in the cdm.
- **tableName**: Name of the table to intersect with. Options: visit_occurrence, condition_occurrence, drug_exposure, procedure_occurrence, device_exposure, measurement, observation, drug_era, condition_era, specimen.
- **indexDate**: Variable in x that contains the date to compute the intersection.
- **censorDate**: Whether to censor overlap events at a specific date or a column date of x.
- **window**: Window to consider events in.
- **targetDate**: Target date in tableName.
- **order**: Which record is considered in case of multiple records (only required for date and days options).
- **nameStyle**: Naming of the added column or columns, should include required parameters.

Value

table with added columns with intersect information.

Examples

```
cdm <- mockPatientProfiles()

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

CDMConnector::cdmDisconnect(cdm = cdm)
```
addTableIntersectDays

Compute time to intersect with an omop table.

Description

Compute time to intersect with an omop table.

Usage

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

Arguments

  x  Table with individuals in the cdm.
  tableName  Name of the table to intersect with. Options: visit_occurrence, condition_occurrence,
              drug_exposure, procedure_occurrence, device_exposure, measurement, observation, drug_era, condition_era, specimen.
  indexDate  Variable in x that contains the date to compute the intersection.
  censorDate  whether to censor overlap events at a specific date or a column date of x.
  window  window to consider events in.
  targetDate  Target date in tableName.
  order  which record is considered in case of multiple records (only required for date
          and days options).
  nameStyle  naming of the added column or columns, should include required parameters.

Value

table with added columns with intersect information.

Examples

cdm <- mockPatientProfiles()

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

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

Description

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

Usage

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

Arguments

- **x**: Table with individuals in the cdm.
- **tableName**: Name of the table to intersect with. Options: visit_occurrence, condition_occurrence, drug_exposure, procedure_occurrence, device_exposure, measurement, observation, drug_era, condition_era, specimen.
- **field**: The columns from the table in tableName to intersect over. For example, if the user uses visit_occurrence in tableName then for field the possible options include visit_occurrence_id, visit_concept_id, visit_type_concept_id.
- **indexDate**: Variable in x that contains the date to compute the intersection.
- **censorDate**: whether to censor overlap events at a specific date or a column date of x.
- **window**: window to consider events in when intersecting with the chosen column.
- **targetDate**: The dates in the target columns in tableName that the user may want to restrict to.
**addTableIntersectFlag**

Compute a flag intersect with an omop table.

**Description**

Compute a flag intersect with an omop table.

**Usage**

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

**Arguments**

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

**Value**

table with added columns with intersect information.

**Examples**

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

indexDate  Variable in x that contains the date to compute the intersection.
censorDate whether to censor overlap events at a specific date or a column date of x.
window    window to consider events in.
targetStartDate Column name with start date for comparison.
targetEndDate Column name with end date for comparison.
nameStyle  naming of the added column or columns, should include required parameters.

Value

table with added columns with intersect information.

Examples

cdm <- mockPatientProfiles()

cdm$cohort1 %>%
  addTableIntersectFlag(tableName = "visit_occurrence")
CDMConnector::cdmDisconnect(cdm = cdm)

assertNameStyle  Assert whether a nameStyle contains the needed information.

Description

Assert whether a nameStyle contains the needed information.

Usage

assertNameStyle(nameStyle, values = list(), call = parent.frame())

Arguments

nameStyle  nameStyle object to check.
values     Parameters options that must be contained.
call       An environment for cli functions.

Value

An error if nameStyle is not properly formatted.
### availableEstimates

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

#### Description

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

#### Usage

availableEstimates(variableType = NULL, fullQuantiles = FALSE)

#### Arguments

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

#### Value

A tibble with the available estimates.
availableFunctions

Examples

library(PatientProfiles)

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

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

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("integer")
availableFunctions("date")
availableFunctions("categorical")
availableFunctions("logical")
**endDateColumn**  
*Get the name of the end date column for a certain table in the cdm*

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

**Usage**
```
endDateColumn(tableName)
```

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

**Value**
Name of the end date column in that table.

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

---

**formatCharacteristics**  
*Format a summarised_characteristics object into a visual table.*

**Description**
`
' r lifecycle::badge("deprecated")'
`

**Usage**
```
formatCharacteristics(
    result,
    type = "gt",
    splitStrata = TRUE,
    format = c("N (%)", "<count> (<percentage>%)", N = "<count>",
               "<median> [<q25> - <q75>]", "<mean> (<sd>)", range = "<min> to <max>",
               cdmName = TRUE,
               cohortName = TRUE,
               style = "default",
               minCellCount = 5,
               .options = list()
)
```
Arguments

result A summarised_characteristics object.
type Type of desired formatted table, possibilities: "gt", "flextable", "tibble".
splitStrata Whether or not to split the strata, default is True.
format The columns that the user wishes to see for the formatted table.
cdmName Whether or not to display the cdm name, default is TRUE.
cohortName Weather or not to display the cohort name, default is TRUE.
style The style of the table output.
minCellCount Default is 5, meaning results that are more than 0 but less than 5 will not be reported.
$options See optionsTableCharacteristics() for default values.

Value

A tibble with a tidy version of the summarised_characteristics object.

Examples

library(PatientProfiles)
cdm <- mockPatientProfiles()
cdm$cohort1 |>
  summariseCharacteristics()
CDMConnector::cdmDisconnect(cdm = cdm)

---

gtCharacteristics Create a gt table from a summarisedCharacteristics object.

Description

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

Usage

gtCharacteristics(
  summarisedCharacteristics,
  pivotWide = c("CDM Name", "Group", "Strata"),
  format = c("N (\%) = \text{count (percentage\%)}", "median [\text{min}; q25 - q75; max]",
            "mean (sd)", "median [q25 - q75]", N = "\text{count}"),
  keepNotFormatted = TRUE,
)
Arguments

summarisedCharacteristics
  Summary characteristics long table.

pivotWide
  variables to pivot wide.

format
  formats and labels to use.

keepNotFormatted
  Whether to keep not formatted estimate types.

decimals
  Decimals per estimate_type.

decimalMark
  decimal mark.

bigMark
  big mark.

Value

New table in gt format.

---

**gtResult**

Create a gt table from a summary object.

Description

'r lifecycle::badge("deprecated")'

Usage

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 = ","
)
mockPatientProfiles

It creates a mock database for testing PatientProfiles package.

Arguments

- summarisedResult: A SummarisedResult object.
- long: List of variables and specification to long.
- wide: List of variables and specification to wide.
- format: Formats and labels to use.
- keepNotFormatted: Whether to keep not formatted estimate types.
- decimals: Decimals per estimate_type.
- decimalMark: Decimal mark.
- bigMark: Big mark.

Value

A formatted summarisedResult gt object.

Examples

library(PatientProfiles)
cdm <- mockPatientProfiles()
cdm$cohort1 %>%
  summariseCharacteristics(
    ageGroup = list(c(0, 19), c(20, 39), c(40, 59), c(60, 79), c(80, 150))
  )
CDMConnector::cdmDisconnect(cdm = cdm)
condition_occurrence = NULL,
visit_occurrence = NULL,
concept_ancestor = NULL,
person = NULL,
death = 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 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.
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.
optionsTableCharacteristics

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

library(PatientProfiles)
cdm <- mockPatientProfiles()
CDMConnector::cdmDisconnect(cdm = cdm)

---

optionsTableCharacteristics

*Additional arguments for the function* tableCharacteristics.

Description

It provides a list of allowed inputs for .option argument in tableCharacteristics, and their given default values.

Usage

optionsTableCharacteristics()

Value

The default .options named list.

Examples

{
  optionsTableCharacteristics()
}


optionsTableCohortOverlap

*Additional arguments for the function tableCohortOverlap.*

**Description**

It provides a list of allowed inputs for `.option` argument in tableCohortOverlap and their given default value.

**Usage**

```r
optionsTableCohortOverlap()
```

**Value**

The default `.options` named list.

**Examples**

```r
{
  optionsTableCohortOverlap()
}
```

optionsTableCohortTiming

*Additional arguments for the function tableCohortTiming.*

**Description**

It provides a list of allowed inputs for `.option` argument in tableCohortTiming and their given default value.

**Usage**

```r
optionsTableCohortTiming()
```

**Value**

The default `.options` named list.

**Examples**

```r
{
  optionsTableCohortTiming()
}
```
plotCharacteristics

Create a ggplot from the output of summariseCharacteristics. ‘r lifecycle::badge("deprecated")’

Description

Create a ggplot from the output of summariseCharacteristics. ‘r lifecycle::badge("deprecated")’

Usage

plotCharacteristics(
  data,
  xAxis = "variable_name",
  yAxis = "estimate_value",
  plotStyle = "barplot",
  facetVarX = NULL,
  facetVarY = NULL,
  colorVars = NULL,
  vertical_x = FALSE
)

Arguments

data output of summariseCharacteristics.

xAxis what to plot on x axis, default as variable_name column. Has to be a column in data.

yAxis what to plot on y axis, default as estimate_value column. Has to be a column in data. One of the xAxis or yAxis has to be estimate_value.

plotStyle Now allows boxplot or barplot only.

facetVarX column in data to facet by on horizontal axis

facetVarY column in data to facet by on vertical axis

colorVars column in data to color by.

vertical_x whether to display x axis string vertically.

Value

A ggplot.

Examples

library(PatientProfiles)

cdm <- mockPatientProfiles()

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

---

plotCohortIntersect  Plot summariseCohortIntersect output.

Description

'\r lifecycle::badge("deprecated")'

Usage

plotCohortIntersect(
  data,
  xAxis = "estimate_value",
  yAxis = "variable_name",
  plotStyle = "barplot",
  facetVarX = "variable_name",
  facetVarY = c("group_level", "strata_level"),
  colorVars = "variable_level",
  vertical_x = TRUE
)

Arguments

data  output of summariseCohortIntersect
xAxis  what to plot on x axis, default as variable_name column. Has to be a column in
data.
yAxis  what to plot on y axis, default as estimate_value column. Has to be a column in
data. One of the xAxis or yAxis has to be estimate_value.
plotStyle  Now allows boxplot or barplot only.
facetVarX  column in data to facet by on horizontal axis
facetVarY  column in data to facet by on vertical axis
colorVars  column in data to color by.
vertical_x  whether to display x axis string vertically.
library(PatientProfiles)
cdm <- mockPatientProfiles()
results <- summariseCohortIntersect(
  cohort = cdm$cohort1,
  cohortIntersect = list(
    "Medications in the prior year" = list(
      targetCohortTable = "cohort2", value = "flag", window = c(-365, -1)
    )
  )
)

graph <- plotCohortIntersect(results)
CDMConnector::cdmDisconnect(cdm = cdm)
Value

A ggplot.

Examples

```r
library(PatientProfiles)
cdm <- PatientProfiles::mockPatientProfiles()
overlap <- summariseCohortOverlap(cdm$cohort2)
plotCohortOverlap(overlap)
```

plotCohortTiming

Plot summariseCohortTiming results.

Description

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

Usage

```r
plotCohortTiming(
  result,
  plotType = "boxplot",
  facetVarX = "variable_name",
  facetVarY = "group_level",
  colorVars = "group_level",
  timingLabel = "{cohort_name_reference} &&& {cohort_name_comparator}",
  uniqueCombinations = TRUE
)
```

Arguments

- `result` A summariseCohortTiming result.
- `plotType` Type of desired formatted table, possibilities are "boxplot" and "density".
- `facetVarX` Column in data to facet by on horizontal axis
- `facetVarY` Column in data to facet by on vertical axis
- `colorVars` Column names to distinct by colors. default set to group_level
- `timingLabel` A glue expression to identify each plotted cohort overlap.
- `uniqueCombinations` If TRUE, only unique combinations of reference and comparator plots will be plotted.

Value

A ggplot.
plotDemographics

Examples

library(PatientProfiles)
cdm <- PatientProfiles::mockPatientProfiles()
timing <- summariseCohortTiming(cdm$cohort2)
plotCohortTiming(timing)
CDMConnector::cdmDisconnect(cdm = cdm)

plotDemographics  Plot summariseDemographics output.

Description

'r lifecycle::badge("deprecated")'

Usage

plotDemographics(
  data,
  xAxis = "variable_name",
  yAxis = "estimate_value",
  plotStyle = "barplot",
  facetVarX = "variable_name",
  facetVarY = c("group_level", "strata_level"),
  colorVars = "variable_level",
  vertical_x = FALSE
)

Arguments

data output of summariseCharacteristics.

xAxis what to plot on x axis, default as variable_name column. Has to be a column in data.

yAxis what to plot on y axis, default as estimate_value column. Has to be a column in data. One of the xAxis or yAxis has to be estimate_value.

plotStyle Now allows boxplot or barplot only.

facetVarX column in data to facet by on horizontal axis

facetVarY column in data to facet by on vertical axis

colorVars column in data to color by.

vertical_x whether to display x axis string vertically.

Value

A ggplot.
Examples

```r
library(PatientProfiles)
cdm <- mockPatientProfiles()
results <- summariseDemographics(
  cohort = cdm$cohort1,
  ageGroup = list(c(0, 19), c(20, 39), c(40, 59), c(60, 79), c(80, 150))
)
graph <- plotDemographics(results)
CDMConnector::cdmDisconnect(cdm = cdm)
```

Description

'\r lifecycle::badge("deprecated")'

Usage

```r
plotLargeScaleCharacteristics(
  data,
  xAxis = "variable_name",
  yAxis = "estimate_value",
  facetVarX = c("variable_name"),
  facetVarY = c("group_level", "strata_level", "estimate_name"),
  colorVars = "variable_level",
  vertical_x = FALSE
)
```

Arguments

data output of summariseLargeScaleCharacteristics.

xAxis what to plot on x axis, default as variable_name column. Has to be a column in data.

yAxis what to plot on y axis, default as estimate_value column. Has to be a column in data. One of the xAxis or yAxis has to be estimate_value.

facetVarX column in data to facet by on horizontal axis

facetVarY column in data to facet by on vertical axis

colorVars column in data to color by.

vertical_x whether to display x axis string vertically.
plotTableIntersect

Value

A ggplot.

Examples

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

concept <- dplyr::tibble(
  concept_id = c(1125315, 1503328, 1516978, 317009, 378253, 4266367),
  domain_id = NA_character_,
  vocabulary_id = NA_character_,
  concept_class_id = NA_character_,
  concept_code = NA_character_,
  valid_start_date = as.Date("1900-01-01"),
  valid_end_date = as.Date("2099-01-01")
)  
  dplyr::mutate(concept_name = paste0("concept: ", .data$concept_id))
cdm <- CDMConnector::insertTable(cdm, "concept", concept)
results <- cdm$cohort2 />
  summariseLargeScaleCharacteristics(
    episodeInWindow = c("condition_occurrence"),
    minimumFrequency = 0
  )
graphs <- plotLargeScaleCharacteristics(results)
CDMConnector::cdmDisconnect(cdm = cdm)

---

plotTableIntersect  Plot summariseTableIntersect output.

Description

"r lifecycle::badge("deprecated")"

Usage

plotTableIntersect(
  data,
  xAxis = "variable_name",
  yAxis = "estimate_value",
  plotStyle = "boxplot",
  facetVarX = "variable_name",
  facetVarY = c("group_level", "strata_level"),
  colorVars = NULL,
  vertical_x = TRUE
)
sourceConceptIdColumn

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

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

Usage
sourceConceptIdColumn(tableName)

Arguments
tableName Name of the table.

Value
Name of the source_concept_id column in that table.

Examples
library(PatientProfiles)
sourceConceptIdColumn("condition_occurrence")
standardConceptIdColumn

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

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

**Usage**

```r
standardConceptIdColumn(tableName)
```

**Arguments**

- `tableName`
  Name of the table.

**Value**
Name of the concept_id column in that table.

**Examples**

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

---

startDateColumn

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

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

**Usage**

```r
startDateColumn(tableName)
```

**Arguments**

- `tableName`
  Name of the table.

**Value**
Name of the start date column in that table.
summariseCharacteristics

Summarise characteristics of individuals

Examples

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

Description

`r lifecycle::badge("deprecated")`

Usage

```r
summariseCharacteristics(
  cohort,
  cdm = lifecycle::deprecated(),
  strata = list(),
  demographics = TRUE,
  ageGroup = NULL,
  tableIntersect = list(),
  cohortIntersect = list(),
  conceptIntersect = list(),
  otherVariables = character()
)
```

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.
**summariseCohortCounts**

**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(
    "Number visits prior year" = list(
      tableName = "visit_occurrence", value = "count", window = c(-365, -1)
    )
  ),
  cohortIntersect = list(
    "Drugs prior year" = list(
      targetCohortTable = "cohort2", value = "flag", window = c(-365, -1)
    ),
    "Conditions any time prior" = list(
      targetCohortTable = "cohort2", value = "flag", window = c(-Inf, -1)
    )
  )
)

CDMConnector::cdmDisconnect(cdm = cdm)
```

**summariseCohortCounts**  
*Summarise counts for each different cohort. You can add a list of stratifications.*

**Description**

‘r lifecycle::badge("deprecated")’

**Usage**

`summariseCohortCounts(cohort, strata = list())`

**Arguments**

- **cohort**: A cohort in the cdm.
- **strata**: Stratification list.

**Value**

A summary of the number of individuals in each cohort and strata.
summariseCohortIntersect

Examples

cdm <- mockPatientProfiles()
cdm$cohort1 |> addSex() |> summariseCohortCounts(strata = "sex")

summariseCohortIntersect
  Summarise cohort intersection information

Description

‘r lifecycle::badge("deprecated")’

Usage

summariseCohortIntersect(cohort, cohortIntersect, strata = list())

Arguments

cohort A cohort in the cdm.
cohortIntersect The settings for cohort intersection settings.
strata Stratification list.

Value

A summary of the cohort intersection informations.

Examples

library(PatientProfiles)
cdm <- mockPatientProfiles()
summariseCohortIntersect(
  cohort = cdm$cohort1,
  cohortIntersect = list("Medications in the prior year" = list(
    targetCohortTable = "cohort2", value = "flag", window = c(-365, -1)
  )
)
)
CDMConnector::cdmDisconnect(cdm = cdm)
summariseCohortOverlap

Summarise cohort overlap

Description

`r lifecycle::badge("deprecated")`

Usage

`summariseCohortOverlap(cohort, cohortId = NULL, strata = list())`

Arguments

- **cohort**: A cohort table in a cdm reference.
- **cohortId**: Vector of cohort definition ids to include, if NULL, all cohort definition ids will be used.
- **strata**: List of the stratifications within each group to be considered. Must be column names in the cohort table provided.

Value

A summarised result.

Examples

```r
library(PatientProfiles)
cdm <- PatientProfiles::mockPatientProfiles()
results <- summariseCohortOverlap(cdm$cohort2)
CDMConnector::cdmDisconnect(cdm = cdm)
```

summariseCohortTiming

Summarise cohort timing

Description

`r lifecycle::badge("deprecated")`
summariseCohortTiming(cohort,
  cohortId = NULL,
  strata = list(),
  restrictToFirstEntry = TRUE,
  timing = c("min", "q25", "median", "q75", "max"),
  density = FALSE
)

Arguments

- **cohort**: A cohort table in a cdm reference.
- **cohortId**: Vector of cohort definition ids to include, if NULL, all cohort definition ids will be used.
- **strata**: List of the stratifications within each group to be considered. Must be column names in the cohort table provided.
- **restrictToFirstEntry**: If TRUE only an individual's first entry per cohort will be considered. If FALSE all entries per individual will be considered.
- **timing**: Summary statistics for timing.
- **density**: Get data for density plot.

Value

A summarised result.

Examples

```r
library(PatientProfiles)
cdm <- PatientProfiles::mockPatientProfiles()
results <- summariseCohortTiming(cdm$cohort2)
CDMConnector::cdmDisconnect(cdm = cdm)
```

summariseConceptIntersect

*Summarise concept intersect with a cohort_table*

Description

`r lifecycle::badge("deprecated")`
summariseDemographics

Usage

summariseConceptIntersect(cohort, conceptIntersect, strata = list())

Arguments

cohort A cohort in the cdm
conceptIntersect A list of arguments that uses addConceptIntersect function to add variables to summarise.
strata Stratification list

Value

A summary of the concept intersect of the individuals

Description

`r lifecycle::badge("deprecated")`

Usage

summariseDemographics(cohort, strata = list(), ageGroup = NULL)

Arguments

cohort A cohort in the cdm.
strata Stratification list.
ageGroup A list of age groups.

Value

A summary of the demographics of the individuals.

Examples

library(PatientProfiles)

cdm <- mockPatientProfiles()

summariseDemographics(
    cohort = cdm$cohort1,
    ageGroup = list(c(0, 19), c(20, 39), c(40, 59), c(60, 79), c(80, 150))
)

CDMConnector::cdmDisconnect(cdm = cdm)
summariseLargeScaleCharacteristics

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

Description

`r lifecycle::badge("deprecated")`

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,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  includeSource = FALSE,
  minimumFrequency = 0.005,
  excludedCodes = c(0),
  cdm = lifecycle::deprecated()
)
```

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. `eventInWindow` must be provided if `episodeInWindow` is not specified.
- **episodeInWindow**: Tables to characterise the episodes in the window. `episodeInWindow` must be provided if `eventInWindow` is not specified.
- **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`.
- **includeSource**: Whether to include source concepts.
- **minimumFrequency**: Minimum frequency covariates to report.
- **excludedCodes**: Codes excluded.
- **cdm**: A cdm reference.
summariseResult

Value

The output of this function is a `ResultSummary` containing the relevant information.

Examples

```r
library(PatientProfiles)
cdm <- PatientProfiles::mockPatientProfiles()

concept <- dplyr::tibble(
  concept_id = c(1125315, 1503328, 1516978, 317009, 378253, 4266367),
  domain_id = NA_character_,
  vocabulary_id = NA_character_,
  concept_class_id = NA_character_,
  concept_code = NA_character_,
  valid_start_date = as.Date("1900-01-01"),
  valid_end_date = as.Date("2099-01-01")
) %>%
  dplyr::mutate(concept_name = paste0("concept: ", .data$concept_id))
cdm <- CDMConnector::insertTable(cdm, "concept", concept)
results <- cdm$cohort2 %>%
  summariseLargeScaleCharacteristics(
    episodeInWindow = c("condition_occurrence"),
    minimumFrequency = 0
  )
CDMConnector::cdmDisconnect(cdm = cdm)
```

summariseResult

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

Description

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

Usage

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

Arguments

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

Value

A summarised_result object with the summarised data of interest.

Examples

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

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

summariseTableIntersect

*Summarise table intersection information*

Description

`r lifecycle::badge("deprecated")`

Usage

```r
summariseTableIntersect(cohort, tableIntersect = list(), strata = list())
```
Arguments

- cohort: A cohort in the cdm.
- tableIntersect: A list of arguments that uses addTableIntersect function to add variables to summarise.
- strata: Stratification list.

Value

A summary of the table intersections.

tableCharacteristics  Format a summarised_characteristics object into a visual table.

Description

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

Usage

\begin{verbatim}
tableCharacteristics(
  result,
  type = "gt",
  formatEstimateName = c("N (\%) = \langle\text{count}\rangle (<\text{percentage}>\%)", N = \langle\text{count}\rangle",
  "Median [Q25 - Q75] = \langle\text{median}\rangle [\langle\text{q25}\rangle - \langle\text{q75}\rangle]", "[Q05 - Q95] = \langle\text{q05}\rangle - \langle\text{q95}\rangle]",
  "Mean (SD) = \langle\text{mean}\rangle (\langle\text{sd}\rangle)", Range = \langle\text{min}\rangle to \langle\text{max}\rangle",
  header = c("group"),
  split = c("group", "strata"),
  groupColumn = NULL,
  minCellCount = 5,
  excludeColumns = c("result_id", "result_type", "package_name", "package_version", "estimate_type", "additional_name", "additional_level"),
  .options = list()
)
\end{verbatim}

Arguments

- result: A summarised_characteristics object.
- type: Type of desired formatted table, possibilities: "gt", "flextable", "tibble".
- formatEstimateName: Named list of estimate name's to join, sorted by computation order. Indicate estimate_name's between <...>.
- header: A vector containing which elements should go into the header in order. Allowed are: 'cdm_name', 'group', 'strata', 'additional', 'variable', 'estimate', 'settings'.
tableCohortIntersect

split A vector containing the name-level groups to split ("group", "strata", "additional"), or an empty character vector to not split.
groupColumn Column to use as group labels.
minCellCount Counts below which results will be clouded.
excludeColumns Columns to drop from the output table.
.options Named list with additional formatting options. PatientProfiles::optionsTableCharacteristics() shows allowed arguments and their default values.

Value
A table with a formatted version of the summariseCharacteristics result.

Examples

library(PatientProfiles)
cdm <- mockPatientProfiles()
cdm$cohort1 |> summariseCharacteristics() |> tableCharacteristics()
CDMConnector::cdmDisconnect(cdm = cdm)

Description
'\r lifecycle::badge("deprecated")'

Usage

tableCohortIntersect(
  result,
  type = "gt",
  formatEstimateName = c(\'N (%)\' = "<count> (<percentage>%)", \'Median [Q25 - Q75]\' = "<median> [<q25> - <q75>]", \'Mean (SD)\' = "<mean> (<sd>)", Range = "<min> to <max>"),
  header = c("group"),
  split = c("group", "strata"),
  groupColumn = NULL,
  minCellCount = 5,
  excludeColumns = c("result_id", "result_type", "package_name", "package_version", "estimate_type", "additional_name", "additional_level"),
  .options = list()
)
tableCohortIntersect

Arguments

result A result from summariseCohortIntersect.
type Type of desired formatted table, possibilities: "gt", "flextable", "tibble".
formatEstimateName Named list of estimate name’s to join, sorted by computation order. Indicate estimate_name’s between <...>.
split A vector containing the name-level groups to split ("group", "strata", "additional"), or an empty character vector to not split.
groupColumn Column to use as group labels.
minCellCount Counts below which results will be clouded.
excludeColumns Columns to drop from the output table.
.options Named list with additional formatting options. PatientProfiles::optionsTableCharacteristics() shows allowed arguments and their default values.

Value

A table with a formatted version of a summariseCohortIntersect result.

Examples

library(PatientProfiles)
cdm <- mockPatientProfiles()
cdm$cohort1 |>
summariseCohortIntersect(
  cohortIntersect = list("Medications in the prior year" = list(
    targetCohortTable = "cohort2", value = "flag", window = c(-365, -1)
  )
)
) |> 
tableCohortIntersect()

CDMConnector::cdmDisconnect(cdm = cdm)
tableCohortOverlap  

Format a summariseOverlapCohort result into a visual table.

**Description**

```
r lifecycle::badge("deprecated")`
```

**Usage**

```r
tableCohortOverlap(  
  result,  
  type = "gt",  
  formatEstimateName = c("N (%)" = "<count> (<percentage>%)"),  
  header = c("strata"),  
  split = c("group", "strata", "additional"),  
  groupColumn = NULL,  
  minCellCount = 5,  
  excludeColumns = c("result_id", "result_type", "package_name", "package_version",  
    "estimate_type"),  
  .options = list()  
)
```

**Arguments**

- `result`: A summariseOverlapCohort result.
- `type`: Type of desired formatted table, possibilities: "gt", "flextable", "tibble".
- `formatEstimateName`: Named list of estimate name's to join, sorted by computation order. Indicate estimate_name's between `<...>`.
- `header`: A vector containing which elements should go into the header in order. Allowed are: `cdm_name`, `group`, `strata`, `additional`, `variable`, `estimate`, `settings`.
- `split`: A vector containing the name-level groups to split ("group", "strata", "additional"), or an empty character vector to not split.
- `groupColumn`: Column to use as group labels.
- `minCellCount`: Counts below which results will be clouded.
- `excludeColumns`: Columns to drop from the output table.
- `.options`: Named list with additional formatting options. PatientProfiles::optionsTableCohortOverlap() shows allowed arguments and their default values.

**Value**

A formatted table of the summariseOverlapCohort result.
**Examples**

```r
library(PatientProfiles)
cdm <- PatientProfiles::mockPatientProfiles()
overlap <- summariseCohortOverlap(cdm$cohort2)
tableCohortOverlap(overlap)
CDMConnector::cdmDisconnect(cdm = cdm)
```

---

**tableCohortTiming**

Format a summariseCohortTiming result into a visual table.

---

**Description**

`r lifecycle::badge("deprecated")`

**Usage**

```r
tableCohortTiming(
  result,
  type = "gt",
  formatEstimateName = c(N = "<count>", `Median [Q25 - Q75]` =
    "<median> [<q25> - <q75>]", Range = "<min> - <max>"),
  header = c("strata"),
  split = c("group", "strata", "additional"),
  groupColumn = NULL,
  minCellCount = 5,
  excludeColumns = c("result_id", "result_type", "package_name", "package_version",
    "estimate_type", "variable_level"),
  .options = list()
)
```

**Arguments**

- **result** A summariseCohortTiming result
- **type** Type of desired formatted table, possibilities: "gt", "flextable", "tibble".
- **formatEstimateName** Named list of estimate name's to join, sorted by computation order. Indicate estimate_name's between `<...>`.
- **header** A vector containing which elements should go into the header in order. Allowed are: `cdm_name`, `group`, `strata`, `additional`, `variable`, `estimate`, `settings`.
- **split** A vector containing the name-level groups to split ("group", "strata", "additional"), or an empty character vector to not split.
- **groupColumn** Column to use as group labels.
minCellCount  Counts below which results will be clouded.
excludeColumns  Columns to drop from the output table.
.options  named list with additional formatting options. PatientProfiles::optionsTableCohortTiming() shows allowed arguments and their default values.

Value
A formatted table of the summariseCohortTiming result.

Examples

library(PatientProfiles)
cdm <- PatientProfiles::mockPatientProfiles()
timing <- summariseCohortTiming(cdm$cohort2)
tableCohortTiming(timing)
CDMConnector::cdmDisconnect(cdm = cdm)

Description

'\texttt{lifecycle::badge("deprecated")}'

Usage

\begin{verbatim}
tableDemographics(
result,
type = "gt",
formatEstimateName = c("\text{N (\%) = \"count\ (\percentage\%)"}, "\text{Median [Q25 - Q75]} = " <median> [<q25> - <q75>]], "\text{[Q05 - Q95]} = " [<q05> - <q95>]], "\text{Mean (SD)} = " <mean> (<sd>)),
header = c("group"),
split = c("group", "strata"),
groupColumn = NULL,
minCellCount = 5,
excludeColumns = c("result_id", "result_type", "package_name", "package_version",
"estimate_type", "additional_name", "additional_level"),
.options = list()
\)
\end{verbatim}
tableLargeScaleCharacteristics

Arguments

result
  A result from summariseDemographics.

type
  Type of desired formatted table, possibilities: "gt", "flextable", "tibble".

formatEstimateName
  Named list of estimate name’s to join, sorted by computation order. Indicate
  estimate_name’s between <...>.

header
  A vector containing which elements should go into the header in order. Al-
  ‘settings’.

split
  A vector containing the name-level groups to split ("group", "strata", "addi-
  tional"), or an empty character vector to not split.

groupColumn
  Column to use as group labels.

minCellCount
  Counts below which results will be clouded.

excludeColumns
  Columns to drop from the output table.

.options
  Named list with additional formatting options. PatientProfiles::optionsTableCharacteristics()
  shows allowed arguments and their default values.

Value

A table with a formatted version of a summariseDemographics result.

Examples

library(PatientProfiles)

cdm <- mockPatientProfiles()

cdm$cohort1 |> summariseDemographics() |> tableDemographics()

CDMConnector::cdmDisconnect(cdm = cdm)

tableLargeScaleCharacteristics

Format a summarised_large_scale_characteristics object into a visual
table.

Description

‘r lifecycle::badge("deprecated")’
Usage

tableLargeScaleCharacteristics(
  result,
  type = "gt",
  formatEstimateName = c("N (%)" = "<count> (<percentage>%)"),
  splitStrata = TRUE,
  header = c("cdm name", "cohort name", "strata", "window name"),
  topConcepts = 10,
  minCellCount = 5
)

Arguments

result  A summarised_large_scalecharacteristics object.
type Output type ("gt" or "flextable").
formatEstimateName Named list of estimate name's to join, sorted by computation order. Indicate estimate_name's between <...>.
splitStrata Whether to split strata_group and strata_level to multiple columns.
header Specify the headers of the table.
topConcepts Number of concepts to restrict the table.
minCellCount Minimum number of counts to display.

Value

A formatted table.

Examples

```r
## Not run:
library(DBI)
library(duckdb)
library(CDMConnector)

con <- dbConnect(duckdb(), eunomia_dir())
cdm <- cdmFromCon(con = con, cdmSchema = "main", writeSchema = "main")
cdm <- generateConceptCohortSet(
  cdm = cdm,
  conceptSet = list("viral_pharyngitis" = 4112343),
  name = "my_cohort"
)
result <- summariseLargeScaleCharacteristics(
  cohort = cdm$my_cohort,
  eventInWindow = "condition_occurrence",
  episodeInWindow = "drug_exposure"
)
tableLargeScaleCharacteristics(result)
```
Format a summariseTableIntersect result into a visual table.

Description

`r lifecycle::badge("deprecated")`

Usage

```r
etableTableIntersect(
  result,
  type = "gt",
  formatEstimateName = c("N (%)" = "<count> (<percentage>%)", "Median [Q25 - Q75]" = "<median> [<q25> - <q75>]", "Mean (SD)" = "<mean> (<sd>)", Range = "<min> to <max>"),
  header = c("group"),
  split = c("group", "strata"),
  groupColumn = NULL,
  minCellCount = 5,
  excludeColumns = c("result_id", "result_type", "package_name", "package_version", "estimate_type", "variable_level", "additional_name", "additional_level"),
  .options = list()
)
```

Arguments

- `result`: A result from summariseTableIntersect.
- `type`: Type of desired formatted table, possibilities: "gt", "flextable", "tibble".
- `formatEstimateName`: Named list of estimate name’s to join, sorted by computation order. Indicate estimate_name’s between `<...>`.
- `split`: A vector containing the name-level groups to split ("group", "strata", "additional"), or an empty character vector to not split.
- `groupColumn`: Column to use as group labels.
- `minCellCount`: Counts below which results will be clouded.
- `excludeColumns`: Columns to drop from the output table.
- `.options`: Named list with additional formatting options. PatientProfiles::optionsTableCharacteristics() shows allowed arguments and their default values.

Value

A table with a formatted version of a summariseTableIntersect result.
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, 3
addCategories, 5
addCdmName, 6
addCohortIntersect, 6
addCohortIntersectCount, 8
addCohortIntersectDate, 9
addCohortIntersectDays, 10
addCohortIntersectFlag, 12
addCohortName, 13
addConceptIntersect, 14
addConceptIntersectCount, 15
addConceptIntersectDate, 17
addConceptIntersectDays, 18
addConceptIntersectFlag, 19
addDateOfBirth, 21
addDeathDate, 22
addDeathDays, 23
addDeathFlag, 24
addDemographics, 25
addFutureObservation, 27
addInObservation, 28
addIntersect, 29
addLargeScaleCharacteristics, 30
addPriorObservation, 31
addSex, 32
addTableIntersect, 33
addTableIntersectCount, 34
addTableIntersectDate, 35
addTableIntersectDays, 37
addTableIntersectField, 38
addTableIntersectFlag, 39
assertNameStyle, 40
availableEstimates, 41
availableFunctions, 42

endDateColumn, 43
formatCharacteristics, 43
gtCharacteristics, 44
gtResult, 45
mockPatientProfiles, 46

optionsTableCharacteristics, 49
optionsTableCohortOverlap, 50
optionsTableCohortTiming, 50
plotCharacteristics, 51
plotCohortIntersect, 52
plotCohortOverlap, 53
plotCohortTiming, 54
plotDemographics, 55
plotLargeScaleCharacteristics, 56
plotTableIntersect, 57

sourceConceptIdColumn, 58
standardConceptIdColumn, 59
startDateColumn, 59
summariseCharacteristics, 60
summariseCohortCounts, 61
summariseCohortIntersect, 62
summariseCohortOverlap, 63
summariseCohortTiming, 63
summariseConceptIntersect, 64
summariseDemographics, 65
summariseLargeScaleCharacteristics, 66
summariseResult, 67
summariseTableIntersect, 68

tableCharacteristics, 69
tableCohortIntersect, 70
tableCohortOverlap, 72
tableCohortTiming, 73
tableDemographics, 74
tableLargeScaleCharacteristics, 75
tableTableIntersect, 77

variableTypes, 78