Package 'PatientProfiles'

February 22, 2024

Type Package

Title Identify Characteristics of Patients in the OMOP Common Data Model

Version 0.6.1

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

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

License Apache License (>= 2)

Encoding UTF-8

RoxygenNote 7.3.1

Suggests covr, duckdb (>= 0.9.0), testthat (>= 3.1.5), knitr, CodelistGenerator, rmarkdown, glue, odbc, ggplot2, spelling, RPostgres, dbplyr, PaRe, here, magick, plotly, ggraph, DT, cowplot, DiagrammeRsvg, DBI

Imports magrittr, CDMConnector (>= 1.3.0), dplyr, tidyr, checkmate, lubridate, rlang, cli, stringr, gt, omopgenerics (>= 0.0.2), visOmopResults, lifecycle

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 2024-02-21 23:30:07 UTC

R topics documented:

---

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 = lifecycle::deprecated(),
  indexDate = "cohort_start_date",
  ageName = "age",
  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()
```

`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)
    ))
  )
```

---

addCdmName *Add cdm name*

Description

Add cdm name

Usage

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

Arguments

table	Table in the cdm
cdm	A cdm reference object

Value

Table with an extra column with the cdm names

Examples

```
library(PatientProfiles)

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

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 = 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

Examples

```
cdm <- mockPatientProfiles()

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

---

addCohortIntersectCount

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

---

Description

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

Usage

```
addCohortIntersectCount(
  x,
  cdm = 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
nameStyle	naming of the added column or columns, should include required parameters

Value

table with added columns with overlap information

Examples

```
cdm <- mockPatientProfiles()

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 = 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 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

```
cdm <- mockPatientProfiles()

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

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 = 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

```
cdm <- mockPatientProfiles()

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

`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
nameStyle	naming of the added column or columns, should include required parameters

Value

table with added columns with overlap information

Examples

```
cdm <- mockPatientProfiles()

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

---

addCohortName *Add cohort name for each cohort_definition_id*

Description

Add cohort name for each cohort_definition_id

Usage

```
addCohortName(cohort)
```

Arguments

cohort	cohort to which add the cohort name

Value

cohort with an extra column with the cohort names

Examples

```
library(PatientProfiles)

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

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

```
addConceptIntersect(
  x,
  conceptSet,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetStartDate = "cohort_start_date",
  targetEndDate = "cohort_end_date",
  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 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

```
library(PatientProfiles)
library(CodelistGenerator)

cdm <- mockPatientProfiles()
# 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

Examples

```
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

```
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

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.
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

Value

table with added columns with overlap information

Examples

```
library(PatientProfiles)
library(CodelistGenerator)

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

---

addConceptIntersectFlag

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

---

Description

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

Usage

```
addConceptIntersectFlag(
  x,
  conceptSet,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  targetStartDate = "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

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

Arguments

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

Value

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

Examples

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

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",
  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.

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

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

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

---

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"
)
```

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

Value

cohort table with added column containing future observation of the individuals

Examples

```
cdm <- mockPatientProfiles()

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

---

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

Description

Indicate if a certain record is within the observation period

Usage

```
addInObservation(
  x,
  cdm = lifecycle::deprecated(),
  indexDate = "cohort_start_date",
  name = "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.
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

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

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

Description

It creates columns to indicate overlap information between two tables

Usage

```
addIntersect(
  x,
  cdm = lifecycle::deprecated(),
  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
`cdm`	A cdm_reference object.
`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

```
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.
censorDate	whether to censor overlap events at a specific date or a column date of x
minimumFrequency	Minimum frequency covariates to report.
excludedCodes	Codes excluded.

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

Usage

```
addPriorObservation(
  x,
  cdm = lifecycle::deprecated(),
  indexDate = "cohort_start_date",
  priorObservationName = "prior_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 prior observation.
priorObservationName	name of the new column to be added

Value

cohort table with added column containing prior observation of the individuals

Examples

```
cdm <- mockPatientProfiles()

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

---

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()
```

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",
  overlap = TRUE,
  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.
window	window to consider events in.
order	which record is considered in case of multiple records (only required for date and days options).
overlap	Whether to consider end date or only end date for the intersection.
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")
```

---

addTableIntersectCount

Compute number of intersect with an omop table.

Description

Compute number of intersect with an omop table.

Usage

```
addTableIntersectCount(
  x,
  tableName,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  overlap = TRUE,
  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.
overlap	Whether to consider end date or only end date for the intersection.
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")
```

---

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")
```

---

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

Obtain a column's value of the intersect with an omop table,

Description

Obtain a column's value of the intersect with an omop table,

Usage

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

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	Other columns from the table to intersect.
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 %>%
  addTableIntersectField(
    tableName = "visit_occurrence",
    field = "visit_concept_id",
    order = "last",
    window = c(-Inf, -1)
  )
```

---

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

Description

Compute a flag intersect with an omop table.

Usage

```
addTableIntersectFlag(
  x,
  tableName,
  indexDate = "cohort_start_date",
  censorDate = NULL,
  window = list(c(0, Inf)),
  overlap = TRUE,
  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.
overlap	Whether to consider end date or only end date for the intersection..
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")
```

---

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("date")
availableFunctions("binary")
availableFunctions("categorical")
```

---

detectVariables *Detect automatically variables with a certain classification*

Description

Detect automatically variables with a certain classification

Usage

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

Arguments

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

Value

Variables in x with the desired classification

Examples

```
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")
```

---

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

Format a summarised_characteristics object into a visual table.

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	.
format	.
cdmName	.
cohortName	.
style	.
minCellCount	.
.options	.

Value

A tibble with a tidy version of the summarised_characteristics object.

Examples

```
library(PatientProfiles)

cdm <- mockPatientProfiles()
```

```
cdm$cohort1 |>
  summariseCharacteristics() |>
  formatCharacteristics()
```

---

gtCharacteristics *Create a gt table from a summarisedCharacteristics object.*

Description

‘r lifecycle::badge("deprecated")‘

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	Whether to keep not formatted 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)
    )
  )
)
```

---

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 = ","
)
```

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))
  )
```

---

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

Description

It creates a mock database for testing PatientProfiles package

Usage

```
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 an details to create the cdm mock object
drug_exposure	default null user can define its own table
drug_strength	default null user can define its own table
observation_period	default null user can define its own table
condition_occurrence	default null user can define its own table
visit_occurrence	default null user can define its own visit_occurrence table
concept_ancestor	the concept ancestor table
person	default null user can define its own table
cohort1	cohort table for test to run in getindication
cohort2	cohort table for test to run in getindication
drug_concept_id_size	number of unique drug concept id

`ancestor_concept_id_size`
: the size of concept ancestor table

`condition_concept_id_size`
: number of unique row in the condition concept table

`visit_concept_id_size`
: number of unique visit concept id

`visit_occurrence_id_size`
: number of unique visit occurrence id

`ingredient_concept_id_size`
: number of unique drug ingredient concept id

`drug_exposure_size`
: number of unique drug exposure

`patient_size`
: number of unique patient

`min_drug_exposure_start_date`
: user define minimum drug exposure start date

`max_drug_exposure_start_date`
: user define maximum drug exposure start date

`earliest_date_of_birth`
: the earliest date of birth of patient in person table format "dd-mm-yyyy"

`latest_date_of_birth`
: the latest date of birth for patient in person table format "dd-mm-yyyy"

`earliest_observation_start_date`
: the earliest observation start date for patient format "dd-mm-yyyy"

`latest_observation_start_date`
: the latest observation start date for patient format "dd-mm-yyyy"

`min_days_to_observation_end`
: the minimum number of days of the observational integer

`max_days_to_observation_end`
: the maximum number of days of the observation period integer

`earliest_condition_start_date`
: the earliest condition start date for patient format "dd-mm-yyyy"

`latest_condition_start_date`
: the latest condition start date for patient format "dd-mm-yyyy"

`min_days_to_condition_end`
: the minimum number of days of the condition integer

`max_days_to_condition_end`
: the maximum number of days of the condition integer

`earliest_visit_start_date`
: the earliest visit start date for patient format "dd-mm-yyyy"

`latest_visit_start_date`
: the latest visit start date for patient format "dd-mm-yyyy"

`min_days_to_visit_end`
: the minimum number of days of the visit integer

`max_days_to_visit_end`
: the maximum number of days of the visit integer

`seed`
: seed

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

Value

cdm of the mock database following user's specifications

Examples

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

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

Description

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

Usage

```
sourceConceptIdColumn(tableName)
```

Arguments

`tableName`	Name of the table

Value

Name of the source_concept_id column in that table

Examples

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

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

Description

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

Usage

```
standardConceptIdColumn(tableName)
```

Arguments

`tableName`	Name of the table

Value

Name of the concept_id column in that table

Examples

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

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

Description

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

Usage

```
startDateColumn(tableName)
```

Arguments

`tableName`	Name of the table

Value

Name of the start date column in that table

Examples

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

`summariseCharacteristics` *Summarise characteristics of individuals*

Description

Summarise characteristics of individuals

Usage

```
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.

Value

A summary of the characteristics of the individuals

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(
    tableName = "visit_occurrence", value = "count", window = c(-365, -1)
  ),
  cohortIntersect = list(
    targetCohortTable = "cohort2", value = "flag", window = c(-365, -1)
  )
)
```

---

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

```
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 = NULL,
  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.

Value

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

---

summariseResult *Summarise the characteristics of different individuals*

Description

Summarise the characteristics of different individuals

Usage

```
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"))
)
```

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.

Value

Table that summarises the characteristics of the individual.

Examples

```
library(PatientProfiles)
library(dplyr)

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

---

tidyCharacteristics *A tidy implementation of the summarised_characteristics object.*

Description

A tidy implementation of the summarised_characteristics object.

Usage

```
tidyCharacteristics(result, minCellCount = 5)
```

Arguments

result	A summarised_characteristics object.
minCellCount	Minimum number of counts to report.

Value

A tibble with a tidy version of the summarised_characteristics object.

Examples

```
library(PatientProfiles)

cdm <- mockPatientProfiles()

cdm$cohort1 |>
  summariseCharacteristics() |>
  tidyCharacteristics()
```

---

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

Description

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

Usage

```
variableTypes(table)
```

Arguments

table	Tibble

Value

Tibble with the variables type and classification

Examples

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

Index