Package 'CohortCharacteristics'

March 27, 2025

Type Package

Title Summarise and Visualise Characteristics of Patients in the OMOP CDM

Version 0.5.1

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

Description Summarise and visualise the characteristics of patients in data mapped to the Observational Medical Outcomes Partnership (OMOP) common data model (CDM).

License Apache License (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

Imports CDMConnector (>= 1.6.0), dplyr, tidyr, rlang, cli, stringr, omopgenerics (>= 0.4.0), PatientProfiles (>= 1.3.1), snakecase, lifecycle, purrr

URL https://darwin-eu.github.io/CohortCharacteristics/

BugReports https://github.com/darwin-eu/CohortCharacteristics/issues

Language en-US

Depends R (>= 4.1)

Config/testthat/edition 3

Config/testthat/parallel true

VignetteBuilder knitr

Suggests CodelistGenerator, visOmopResults (>= 1.0.0), CohortConstructor, covr, DBI, dbplyr, DiagrammeR, DiagrammeRsvg, DrugUtilisation, duckdb (>= 1.0.0), flextable, ggplot2, ggpubr, glue, grid, gt, here, Hmisc, htmltools, knitr, odbc, omock, plotly, png, rmarkdown, RPostgres, rsvg, scales, spelling, testthat (>= 3.1.5), tictoc, withr

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 [ctb] (<https://orcid.org/0000-0002-9517-8834>),
Kim Lopez-Guell [aut] (<https://orcid.org/0000-0002-8462-8668>),
Edward Burn [aut] (<https://orcid.org/0000-0002-9286-1128>),
Nuria Mercade-Besora [aut] (<https://orcid.org/0009-0006-7948-3747>),
Marta Alcalde [aut] (<https://orcid.org/0009-0002-4405-1814>)

Repository CRAN

Date/Publication 2025-03-27 18:40:07 UTC

Contents

---

availablePlotColumns *Available columns to use in facet and colour arguments in plot functions.*

Description

Available columns to use in `facet` and `colour` arguments in plot functions.

Usage

```
availablePlotColumns(result)
```

Arguments

`result`	A summarised_result object.

Value

Character vector with the available columns.

Examples

```
{
cdm <- mockCohortCharacteristics()

result <- summariseCharacteristics(cdm$cohort1)

availablePlotColumns(result)

mockDisconnect(cdm)
}
```

`availableTableColumns` *Available columns to use in* `header`, `groupColumn` *and* `hide` *arguments in table functions.*

Description

Available columns to use in `header`, `groupColumn` and `hide` arguments in table functions.

Usage

```
availableTableColumns(result)
```

Arguments

`result`	A summarised_result object.

Value

Character vector with the available columns.

Examples

```
{
cdm <- mockCohortCharacteristics()

result <- summariseCharacteristics(cdm$cohort1)

availableTableColumns(result)

mockDisconnect(cdm)
}
```

benchmarkCohortCharacteristics

Benchmark the main functions of CohortCharacteristics package.

Description

Benchmark the main functions of CohortCharacteristics package.

Usage

```
benchmarkCohortCharacteristics(
  cohort,
  analysis = c("count", "attrition", "characteristics", "overlap", "timing",
    "large scale characteristics")
)
```

Arguments

`cohort`	A cohort_table from a cdm_reference.
`analysis`	Set of analysis to perform, must be a subset of: "count", "attrition", "characteristics", "overlap", "timing" and "large scale characteristics".

Value

A summarised_result object.

Examples

```
## Not run:
CDMConnector::requireEunomia()
con <- duckdb::dbConnect(duckdb::duckdb(), CDMConnector::eunomiaDir())
cdm <- CDMConnector::cdmFromCon(
  con = con, cdmSchema = "main", writeSchema = "main"
)

cdm <- CDMConnector::generateConceptCohortSet(
  cdm = cdm,
  conceptSet = list(sinusitis = 40481087, pharyngitis = 4112343),
  name = "my_cohort"
)

benchmarkCohortCharacteristics(cdm$my_cohort)


## End(Not run)
```

---

mockCohortCharacteristics

It creates a mock database for testing CohortCharacteristics package

---

Description

It creates a mock database for testing CohortCharacteristics package

Usage

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

Arguments

`con` A DBI connection to create the cdm mock object.

`writeSchema` Name of an schema on the same connection with writing permissions.

`numberIndividuals` Number of individuals to create in the cdm reference.

`...` User self defined tables to put in cdm, it can input as many as the user want.

`seed` A number to set the seed. If NULL seed is not used.

Value

A mock cdm_reference object created following user's specifications.

Examples

```
library(CohortCharacteristics)
library(CDMConnector)

cdm <- mockCohortCharacteristics()
```

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

plotCharacteristics *Create a ggplot from the output of summariseCharacteristics.*

Description

[Experimental]

Usage

```
plotCharacteristics(
  result,
  plotType = "barplot",
  facet = NULL,
  colour = NULL,
  plotStyle = lifecycle::deprecated()
)
```

Arguments

`result`	A summarised_result object.
`plotType`	Either `barplot`, `scatterplot` or `boxplot`. If `barplot` or `scatterplot` subset to just one estimate.
`facet`	Columns to facet by. See options with `availablePlotColumns(result)`. Formula is also allowed to specify rows and columns.
`colour`	Columns to color by. See options with `availablePlotColumns(result)`.
`plotStyle`	deprecated.

Value

A ggplot.

Examples

```
library(CohortCharacteristics)
library(dplyr, warn.conflicts = FALSE)

cdm <- mockCohortCharacteristics()

results <- summariseCharacteristics(
  cohort = cdm$cohort1,
  ageGroup = list(c(0, 19), c(20, 39), c(40, 59), c(60, 79), c(80, 150)),
  tableIntersectCount = list(
    tableName = "visit_occurrence", window = c(-365, -1)
    ),
    cohortIntersectFlag = list(
      targetCohortTable = "cohort2", window = c(-365, -1)
    )
  )

results |>
  filter(
    variable_name == "Cohort2 flag -365 to -1", estimate_name == "percentage"
  ) |>
  plotCharacteristics(
    plotType = "barplot",
    colour = "variable_level",
    facet = c("cdm_name", "cohort_name")
  )

results |>
  filter(variable_name == "Age", estimate_name == "mean") |>
  plotCharacteristics(
    plotType = "scatterplot",
    facet = "cdm_name"
  )

results |>
  filter(variable_name == "Age", group_level == "cohort_1") |>
  plotCharacteristics(
    plotType = "boxplot",
    facet = "cdm_name",
    colour = "cohort_name"
  )

mockDisconnect(cdm)
```

`plotCohortAttrition` *create a ggplot from the output of summariseLargeScaleCharacteristics.*

Description

[Experimental]

Usage

```
plotCohortAttrition(
  result,
  show = c("subjects", "records"),
  type = "htmlwidget",
  cohortId = lifecycle::deprecated()
)
```

Arguments

`result` A summarised_result object.

`show` Which variables to show in the attrition plot, it can be ’subjects’, ’records’ or both.

`type` type of the output, it can either be: ’htmlwidget’, ’png’, or ’DiagrammeR’.

`cohortId` deprecated.

Value

A `grViz` visualisation.

Examples

```
library(CohortCharacteristics)
library(omopgenerics)
library(dplyr, warn.conflicts = FALSE)

cdm <- mockCohortCharacteristics(numberIndividuals = 1000)

cdm[[”cohort1”]] <- cdm[[”cohort1”]] |>
  filter(year(cohort_start_date) >= 2000) |>
  recordCohortAttrition(”Restrict to cohort_start_date >= 2000”) |>
  filter(year(cohort_end_date) < 2020) |>
  recordCohortAttrition(”Restrict to cohort_end_date < 2020”) |>
  compute(temporary = FALSE, name = ”cohort1”)

result <- summariseCohortAttrition(cdm$cohort1)

result |>
  filter(group_level == ”cohort_2”) |>
  plotCohortAttrition()

mockDisconnect(cdm)
```

`plotCohortCount` *Plot the result of summariseCohortCount.*

Description

[Experimental]

Usage

```
plotCohortCount(result, x = NULL, facet = c(”cdm_name”), colour = NULL)
```

Arguments

`result` A summarised_result object.

`x` Variables to use in x axis.

`facet` Columns to facet by. See options with `availablePlotColumns(result)`. Formula is also allowed to specify rows and columns.

`colour` Columns to color by. See options with `availablePlotColumns(result)`.

Value

A ggplot.

Examples

```
library(CohortCharacteristics)
library(PatientProfiles)
library(dplyr, warn.conflicts = FALSE)

cdm <- mockCohortCharacteristics(numberIndividuals = 100)

counts <- cdm$cohort2 |>
  addSex() |>
  addAge(ageGroup = list(c(0, 29), c(30, 59), c(60, Inf))) |>
  summariseCohortCount(strata = list("age_group", "sex", c("age_group", "sex"))) |>
  filter(variable_name == "Number subjects")

counts |>
  plotCohortCount(
    x = "sex",
    facet = cohort_name ~ age_group,
    colour = "sex"
  )

mockDisconnect(cdm)
```

`plotCohortOverlap` *Plot the result of summariseCohortOverlap.*

Description

[Experimental]

Usage

```
plotCohortOverlap(
  result,
  uniqueCombinations = TRUE,
  y = NULL,
  facet = c("cdm_name", "cohort_name_reference"),
  colour = "variable_name",
  .options = lifecycle::deprecated()
)
```

Arguments

`result`	A summarised_result object.
`uniqueCombinations`	Whether to restrict to unique reference and comparator comparisons.
`y`	Variables to use in y axis, if NULL all variables not present in facet are used.
`facet`	Columns to facet by. See options with `availablePlotColumns(result)`. Formula is also allowed to specify rows and columns.
`colour`	Columns to color by. See options with `availablePlotColumns(result)`.
`.options`	deprecated.

Value

A ggplot.

Examples

```
library(CohortCharacteristics)

cdm <- mockCohortCharacteristics()

overlap <- summariseCohortOverlap(cdm$cohort2)

plotCohortOverlap(overlap, uniqueCombinations = FALSE)

mockDisconnect(cdm)
```

`plotCohortTiming` *Plot summariseCohortTiming results.*

Description

[Experimental]

Usage

```
plotCohortTiming(
  result,
  plotType = "boxplot",
  timeScale = "days",
  uniqueCombinations = TRUE,
  facet = c("cdm_name", "cohort_name_reference"),
  colour = c("cohort_name_comparator")
)
```

Arguments

`result` A summarised_result object.

`plotType` Type of desired formatted table, possibilities are "boxplot" and "densityplot".

`timeScale` Time scale to show, it can be "days" or "years".

`uniqueCombinations` Whether to restrict to unique reference and comparator comparisons.

`facet` Columns to facet by. See options with `availablePlotColumns(result)`. Formula is also allowed to specify rows and columns.

`colour` Columns to color by. See options with `availablePlotColumns(result)`.

Value

A ggplot.

Examples

```
## Not run:
library(CohortCharacteristics)
library(duckdb)
library(CDMConnector)
library(DrugUtilisation)

con <- dbConnect(duckdb(), eunomiaDir())
cdm <- cdmFromCon(con, cdmSchem = "main", writeSchema = "main")

cdm <- generateIngredientCohortSet(
  cdm = cdm,
  name = "my_cohort",
  ingredient = c("acetaminophen", "morphine", "warfarin")
)

timings <- summariseCohortTiming(cdm$my_cohort)

plotCohortTiming(
  timings,
  timeScale = "years",
  uniqueCombinations = FALSE,
  facet = c("cdm_name", "cohort_name_reference"),
  colour = c("cohort_name_comparator")
)

plotCohortTiming(
  timings,
  plotType = "densityplot",
  timeScale = "years",
  uniqueCombinations = FALSE,
  facet = c("cdm_name", "cohort_name_reference"),
  colour = c("cohort_name_comparator")
)

cdmDisconnect(cdm)

## End(Not run)
```

plotComparedLargeScaleCharacteristics

create a ggplot from the output of summariseLargeScaleCharacteristics.

Description

[Experimental]

Usage

```
plotComparedLargeScaleCharacteristics(
  result,
  reference,
  missings = 0,
  facet = NULL,
  colour = NULL
)
```

Arguments

`result` A summarised_result object.

`reference` A named character to set up the reference.

`missings` Value to replace the missing value with. If NULL missing values will be eliminated.

`facet` Columns to facet by. See options with `availablePlotColumns(result)`. Formula is also allowed to specify rows and columns.

`colour` Columns to color by. See options with `availablePlotColumns(result)`.

Value

A ggplot.

Examples

```
## Not run:
library(CohortCharacteristics)
library(duckdb)
library(CDMConnector)
library(DrugUtilisation)
library(plotly, warn.conflicts = FALSE)

con <- dbConnect(duckdb(), eunomiaDir())
cdm <- cdmFromCon(con, cdmSchem = "main", writeSchema = "main")

cdm <- generateIngredientCohortSet(
  cdm = cdm, name = "my_cohort", ingredient = "acetaminophen"
)

resultsLsc <- cdm$my_cohort |>
  summariseLargeScaleCharacteristics(
    window = list(c(-365, -1), c(1, 365)),
    eventInWindow = "condition_occurrence"
  )

resultsLsc |>
  plotComparedLargeScaleCharacteristics(
    reference = c(variable_level = "-365 to -1"),
    colour = "variable_name",
    missings = NULL
  ) |>
  ggplotly()

cdmDisconnect(cdm)

## End(Not run)
```

plotLargeScaleCharacteristics

create a ggplot from the output of summariseLargeScaleCharacteristics.

Description

[Experimental]

Usage

```
plotLargeScaleCharacteristics(
  result,
  facet = c("cdm_name", "cohort_name"),
  colour = "variable_level"
)
```

Arguments

result	A summarised_result object.
facet	Columns to facet by. See options with availablePlotColumns(result). Formula is also allowed to specify rows and columns.
colour	Columns to color by. See options with availablePlotColumns(result).

Value

A ggplot2 object.

Examples

```
## Not run:
library(CohortCharacteristics)
library(duckdb)
library(CDMConnector)
library(DrugUtilisation)

con <- dbConnect(duckdb(), eunomiaDir())
cdm <- cdmFromCon(con, cdmSchem = "main", writeSchema = "main")

cdm <- generateIngredientCohortSet(
  cdm = cdm, name = "my_cohort", ingredient = "acetaminophen"
)

resultsLsc <- cdm$my_cohort |>
  summariseLargeScaleCharacteristics(
    window = list(c(-365, -1), c(1, 365)),
    eventInWindow = "condition_occurrence"
  )

resultsLsc |>
  plotLargeScaleCharacteristics(
    facet = c("cdm_name", "cohort_name"),
    colour = "variable_level"
  )

cdmDisconnect(cdm)

## End(Not run)
```

`summariseCharacteristics` *Summarise characteristics of cohorts in a cohort table*

Description

Summarise characteristics of cohorts in a cohort table

Usage

```
summariseCharacteristics(
  cohort,
  cohortId = NULL,
  strata = list(),
  counts = TRUE,
  demographics = TRUE,
  ageGroup = NULL,
  tableIntersectFlag = list(),
  tableIntersectCount = list(),
  tableIntersectDate = list(),
  tableIntersectDays = list(),
  cohortIntersectFlag = list(),
  cohortIntersectCount = list(),
  cohortIntersectDate = list(),
  cohortIntersectDays = list(),
  conceptIntersectFlag = list(),
  conceptIntersectCount = list(),
  conceptIntersectDate = list(),
  conceptIntersectDays = list(),
  otherVariables = character(),
  estimates = list(),
  weights = NULL,
  otherVariablesEstimates = lifecycle::deprecated()
)
```

Arguments

`cohort`	A cohort_table object.
`cohortId`	A cohort definition id to restrict by. If NULL, all cohorts will be included.
`strata`	A list of variables to stratify results. These variables must have been added as additional columns in the cohort table.
`counts`	TRUE or FALSE. If TRUE, record and person counts will be produced.
`demographics`	TRUE or FALSE. If TRUE, patient demographics (cohort start date, cohort end date, age, sex, prior observation, and future observation will be summarised).
`ageGroup`	A list of age groups to stratify results by.

`tableIntersectFlag`
A list of arguments that uses PatientProfiles::addTableIntersectFlag() to add variables to summarise.

`tableIntersectCount`
A list of arguments that uses PatientProfiles::addTableIntersectCount() to add variables to summarise.

`tableIntersectDate`
A list of arguments that uses PatientProfiles::addTableIntersectDate() to add variables to summarise.

`tableIntersectDays`
A list of arguments that uses PatientProfiles::addTableIntersectDays() to add variables to summarise.

`cohortIntersectFlag`
A list of arguments that uses PatientProfiles::addCohortIntersectFlag() to add variables to summarise.

`cohortIntersectCount`
A list of arguments that uses PatientProfiles::addCohortIntersectCount() to add variables to summarise.

`cohortIntersectDate`
A list of arguments that uses PatientProfiles::addCohortIntersectDate() to add variables to summarise.

`cohortIntersectDays`
A list of arguments that uses PatientProfiles::addCohortIntersectDays() to add variables to summarise.

`conceptIntersectFlag`
A list of arguments that uses PatientProfiles::addConceptIntersectFlag() to add variables to summarise.

`conceptIntersectCount`
A list of arguments that uses PatientProfiles::addConceptIntersectCount() to add variables to summarise.

`conceptIntersectDate`
A list of arguments that uses PatientProfiles::addConceptIntersectDate() to add variables to summarise.

`conceptIntersectDays`
A list of arguments that uses PatientProfiles::addConceptIntersectDays() to add variables to summarise.

`otherVariables` Other variables contained in cohort that you want to be summarised.

`estimates` To modify the default estimates for a variable. By default: 'min', 'q25', 'median', 'q75', 'max' for "date", "numeric" and "integer" variables ("numeric" and "integer" also use 'mean' and 'sd' estimates). 'count' and 'percentage' for "categorical" and "binary". You have to provide them as a list: `list(age = c("median", "density"))`. You can also use 'date', 'numeric', 'integer', 'binary', 'categorical', 'demographics', 'intersect', 'other', 'table_intersect_count', ...

`weights` Column in cohort that points to weights of each individual.

`otherVariablesEstimates`
deprecated.

Value

A summary of the characteristics of the cohorts in the cohort table.

Examples

```
library(dplyr, warn.conflicts = FALSE)
library(CohortCharacteristics)
library(PatientProfiles)

cdm <- mockCohortCharacteristics()

cdm$cohort1 |>
  addSex() |>
  addAge(
    ageGroup = list(c(0, 40), c(41, 150))
  ) |>
  summariseCharacteristics(
    strata = list("sex", "age_group"),
    cohortIntersectFlag = list(
      "Cohort 2 Flag" = list(
        targetCohortTable = "cohort2", window = c(-365, 0)
      )
    ),
    cohortIntersectCount = list(
      "Cohort 2 Count" = list(
        targetCohortTable = "cohort2", window = c(-365, 0)
      )
    )
  ) |>
  glimpse()

mockDisconnect(cdm)
```

summariseCohortAttrition

Summarise attrition associated with cohorts in a cohort table

Description

Summarise attrition associated with cohorts in a cohort table

Usage

```
summariseCohortAttrition(cohort, cohortId = NULL)
```

Arguments

cohort	A cohort_table object.
cohortId	A cohort definition id to restrict by. If NULL, all cohorts will be included.

Value

A summary of the attrition for the cohorts in the cohort table.

Examples

```
library(CohortCharacteristics)
library(dplyr, warn.conflicts = FALSE)

cdm <- mockCohortCharacteristics()

summariseCohortAttrition(cohort = cdm$cohort1) |>
  glimpse()

mockDisconnect(cdm)
```

---

summariseCohortCount *Summarise counts for cohorts in a cohort table*

Description

Summarise counts for cohorts in a cohort table

Usage

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

Arguments

cohort	A cohort_table object.
cohortId	A cohort definition id to restrict by. If NULL, all cohorts will be included.
strata	A list of variables to stratify results. These variables must have been added as additional columns in the cohort table.

Value

A summary of counts of the cohorts in the cohort table.

Examples

```
library(CohortCharacteristics)
library(dplyr, warn.conflicts = FALSE)

cdm <- mockCohortCharacteristics()

summariseCohortCount(cohort = cdm$cohort1) |>
  glimpse()

mockDisconnect(cdm)
```

summariseCohortOverlap *Summarise overlap between cohorts in a cohort table*

Description

Summarise overlap between cohorts in a cohort table

Usage

```
summariseCohortOverlap(
  cohort,
  cohortId = NULL,
  strata = list(),
  overlapBy = "subject_id"
)
```

Arguments

`cohort`	A cohort_table object.
`cohortId`	A cohort definition id to restrict by. If NULL, all cohorts will be included.
`strata`	A list of variables to stratify results. These variables must have been added as additional columns in the cohort table.
`overlapBy`	Columns in cohort to use as record identifiers.

Value

A summary of overlap between cohorts in the cohort table.

Examples

```
library(CohortCharacteristics)
library(dplyr, warn.conflicts = FALSE)

cdm <- mockCohortCharacteristics()

summariseCohortOverlap(cdm$cohort2) |>
  glimpse()

mockDisconnect(cdm)
```

`summariseCohortTiming` *Summarise timing between entries into cohorts in a cohort table*

Description

Summarise timing between entries into cohorts in a cohort table

Usage

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

Arguments

`cohort` A cohort_table object.

`cohortId` A cohort definition id to restrict by. If NULL, all cohorts will be included.

`strata` A list of variables to stratify results. These variables must have been added as additional columns in the cohort table.

`restrictToFirstEntry` If TRUE only an individual's first entry per cohort will be considered. If FALSE all entries per individual will be considered.

`estimates` Summary statistics to use when summarising timing.

`density` deprecated.

Value

A summary of timing between entries into cohorts in the cohort table.

Examples

```
library(CohortCharacteristics)
library(dplyr, warn.conflicts = FALSE)

cdm <- mockCohortCharacteristics(numberIndividuals = 100)

summariseCohortTiming(cdm$cohort2) |>
  glimpse()

mockDisconnect(cdm)
```

`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,
  cohortId = NULL,
  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)
)
```

Arguments

`cohort` A cohort_table object.

`cohortId` A cohort definition id to restrict by. If NULL, all cohorts will be included.

`strata` A list of variables to stratify results. These variables must have been added as additional columns in the cohort table.

`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 of codes to be reported. If a concept_id has a frequency smaller than `minimumFrequency` in a certain window that estimate will be eliminated from the result object.

`excludedCodes` Codes excluded.

Value

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

Examples

```
## Not run:
library(CohortCharacteristics)
library(duckdb)
library(CDMConnector)
library(DrugUtilisation)
library(dplyr, warn.conflicts = FALSE)

con <- dbConnect(duckdb(), eunomiaDir())
cdm <- cdmFromCon(con, cdmSchem = "main", writeSchema = "main")

cdm <- generateIngredientCohortSet(
  cdm = cdm, name = "my_cohort", ingredient = "acetaminophen"
)

cdm$my_cohort |>
  summariseLargeScaleCharacteristics(
    window = list(c(-365, -1), c(1, 365)),
    eventInWindow = "condition_occurrence"
  ) |>
  glimpse()

cdmDisconnect(cdm)

## End(Not run)
```

`tableCharacteristics` *Format a summarise_characteristics object into a visual table.*

Description

[Experimental]

Usage

```
tableCharacteristics(
  result,
  type = "gt",
  header = c("cdm_name", "cohort_name"),
  groupColumn = character(),
  hide = c(additionalColumns(result), settingsColumns(result))
)
```

Arguments

`result` A summarised_result object.

`type` Type of table. Check supported types with `visOmopResults::tableType()`.

`header` Columns to use as header. See options with `availableTableColumns(result)`.

`groupColumn` Columns to group by. See options with `availableTableColumns(result)`.

`hide` Columns to hide from the visualisation. See options with `availableTableColumns(result)`.

Value

A formatted table.

Examples

```
library(CohortCharacteristics)

cdm <- mockCohortCharacteristics()

result <- summariseCharacteristics(cdm$cohort1)

tableCharacteristics(result)

mockDisconnect(cdm)
```

`tableCohortAttrition` *Create a visual table from the output of summariseCohortAttrition.*

Description

[Experimental]

Usage

```
tableCohortAttrition(
  result,
  type = "gt",
  header = "variable_name",
  groupColumn = c("cdm_name", "cohort_name"),
 hide = c("variable_level", "reason_id", "estimate_name", settingsColumns(result))
)
```

Arguments

result	A summarised_result object.
type	Type of table. Check supported types with `visOmopResults::tableType()`.
header	Columns to use as header. See options with `availableTableColumns(result)`.
groupColumn	Columns to group by. See options with `availableTableColumns(result)`.
hide	Columns to hide from the visualisation. See options with `availableTableColumns(result)`.

Value

A formatted table.

Examples

```
library(CohortCharacteristics)

cdm <- mockCohortCharacteristics()

result <- summariseCohortAttrition(cdm$cohort2)

tableCohortAttrition(result)

mockDisconnect(cdm)
```

tableCohortCount *Format a summarise_characteristics object into a visual table.*

Description

[Experimental]

Usage

```
tableCohortCount(
  result,
  type = "gt",
  header = "cohort_name",
  groupColumn = character(),
  hide = c("variable_level", settingsColumns(result))
)
```

Arguments

`result`	A summarised_result object.
`type`	Type of table. Check supported types with `visOmopResults::tableType()`.
`header`	Columns to use as header. See options with `availableTableColumns(result)`.
`groupColumn`	Columns to group by. See options with `availableTableColumns(result)`.
`hide`	Columns to hide from the visualisation. See options with `availableTableColumns(result)`.

Value

A formatted table.

Examples

```
library(CohortCharacteristics)

cdm <- mockCohortCharacteristics()

result <- summariseCohortCount(cdm$cohort1)

tableCohortCount(result)

mockDisconnect(cdm = cdm)
```

`tableCohortOverlap` *Format a summariseOverlapCohort result into a visual table.*

Description

[Experimental]

Usage

```
tableCohortOverlap(
  result,
  uniqueCombinations = TRUE,
  type = "gt",
  header = c("variable_name"),
  groupColumn = c("cdm_name"),
  hide = c("variable_level", settingsColumns(result))
)
```

Arguments

`result`	A summarised_result object.
`uniqueCombinations`	Whether to restrict to unique reference and comparator comparisons.
`type`	Type of table. Check supported types with `visOmopResults::tableType()`.
`header`	Columns to use as header. See options with `availableTableColumns(result)`.
`groupColumn`	Columns to group by. See options with `availableTableColumns(result)`.
`hide`	Columns to hide from the visualisation. See options with `availableTableColumns(result)`.

Value

A formatted table.

Examples

```
library(CohortCharacteristics)

cdm <- mockCohortCharacteristics()

overlap <- summariseCohortOverlap(cdm$cohort2)

tableCohortOverlap(overlap)

mockDisconnect(cdm = cdm)
```

tableCohortTiming *Format a summariseCohortTiming result into a visual table.*

Description

[Experimental]

Usage

```
tableCohortTiming(
  result,
  timeScale = "days",
  uniqueCombinations = TRUE,
  type = "gt",
  header = strataColumns(result),
  groupColumn = c("cdm_name"),
  hide = c("variable_level", settingsColumns(result))
)
```

Arguments

`result`	A summarised_result object.
`timeScale`	Time scale to show, it can be "days" or "years".
`uniqueCombinations`	Whether to restrict to unique reference and comparator comparisons.
`type`	Type of table. Check supported types with `visOmopResults::tableType()`.
`header`	Columns to use as header. See options with `availableTableColumns(result)`.
`groupColumn`	Columns to group by. See options with `availableTableColumns(result)`.
`hide`	Columns to hide from the visualisation. See options with `availableTableColumns(result)`.

Value

A formatted table.

Examples

```
## Not run:
library(CohortCharacteristics)
library(duckdb)
library(CDMConnector)
library(DrugUtilisation)

con <- dbConnect(duckdb(), eunomiaDir())
cdm <- cdmFromCon(con, cdmSchem = "main", writeSchema = "main")

cdm <- generateIngredientCohortSet(
  cdm = cdm,
  name = "my_cohort",
  ingredient = c("acetaminophen", "morphine", "warfarin")
)

timings <- summariseCohortTiming(cdm$my_cohort)

tableCohortTiming(timings, timeScale = "years")

cdmDisconnect(cdm)

## End(Not run)
```

`tableLargeScaleCharacteristics`

Format a summarise_large_scale_characteristics object into a visual table.

Description

[Experimental]

Usage

```
tableLargeScaleCharacteristics(
  result,
  topConcepts = NULL,
  type = "gt",
  header = c("cdm_name", "cohort_name", strataColumns(result), "variable_level"),
  groupColumn = c("table_name", "type", "analysis"),
  hide = character()
)
```

Arguments

`result`	A summarised_result object.
`topConcepts`	Number of concepts to restrict the table.
`type`	Type of table. Check supported types with `visOmopResults::tableType()`.
`header`	Columns to use as header. See options with `availableTableColumns(result)`.
`groupColumn`	Columns to group by. See options with `availableTableColumns(result)`.
`hide`	Columns to hide from the visualisation. See options with `availableTableColumns(result)`.

Value

A formatted table.

Examples

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

con <- dbConnect(duckdb(), eunomiaDir())
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)

cdmDisconnect(cdm)

## End(Not run)
```

Index