Package 'visOmopResults'

September 24, 2024

Title Graphs and Tables for OMOP Results

Version 0.4.0

Maintainer Núria Mercadé-Besora <nuria.mercadebesora@ndorms.ox.ac.uk>

Description Provides methods to transform omop_result objects into formatted tables and figures, facilitating the visualization of study results working with the Observational Medical Outcomes Partnership (OMOP) Common Data Model.

License Apache License (>= 2)

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

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

Imports cli, dplyr, generics, glue, lifecycle, omopgenerics (>= 0.2.0), purrr, rlang, stringr, tidyr

Suggests covr, flextable (>= 0.9.5), ggplot2, gt, knitr, officer, palmerpenguins, PatientProfiles, plotly, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Martí Català [aut] (<https://orcid.org/0000-0003-3308-9905>), Núria Mercadé-Besora [aut, cre] (<https://orcid.org/0009-0006-7948-3747>), Yuchen Guo [aut] (<https://orcid.org/0000-0002-0847-4855>)

Repository CRAN

Date/Publication 2024-09-23 22:00:02 UTC

Contents

additionalColumns *Identify variables in additional_name column*

Description

Identifies and returns the unique values in additional_name column.

Usage

```
additionalColumns(result)
```

Arguments

result	A tibble.

Value

Unique values of the additional name column.

Examples

```
mockSummarisedResult() |>
  additionalColumns()
```

addSettings *Add settings columns to a* <summarised_result> *object*

Description

Add settings columns to a <summarised_result> object

Usage

```
addSettings(
  result,
  settingsColumns = colnames(settings(result)),
  columns = lifecycle::deprecated()
)
```

Arguments

result	A <summarised_result> object.
settingsColumns	Settings to be added as columns, by default all settings will be added. If NULL or empty character vector, no settings will be added.
columns	**[Deprecated]**

Value

A <summarised_result> object with the added setting columns.

Examples

```
library(visOmopResults)
mockSummarisedResult() |>
  addSettings(settingsColumns = c("result_type"))
```

---

barPlot *Create a bar plot visualisation from a* <summarised_result> *object*

Description

[Experimental]

Usage

```
barPlot(result, x, y, facet = NULL, colour = NULL)
```

Arguments

result	A <summarised_result> object.
x	Column or estimate name that is used as x variable.
y	Column or estimate name that is used as y variable
facet	Variables to facet by, a formula can be provided to specify which variables should be used as rows and which ones as columns.
colour	Columns to use to determine the colors.

Value

A plot object.

Examples

```
result <- mockSummarisedResult() |> dplyr::filter(variable_name == "age")

barPlot(
  result = result,
  x = "cohort_name",
  y = "mean",
  facet = c("age_group", "sex"),
  colour = "sex")
```

boxPlot *Create a box plot visualisation from a* <summarised_result> *object*

Description

[Experimental]

Usage

```
boxPlot(
  result,
  x = NULL,
  lower = "q25",
  middle = "median",
  upper = "q75",
  ymin = "min",
  ymax = "max",
  facet = NULL,
  colour = NULL
)
```

Arguments

result	A <summarised_result> object.
x	Columns to use as x axes.
lower	Estimate name for the lower quantile of the box.
middle	Estimate name for the middle line of the box.
upper	Estimate name for the upper quantile of the box.
ymin	Estimate name for the lower limit of the bars.
ymax	Estimate name for the upper limit of the bars.
facet	Variables to facet by, a formula can be provided to specify which variables should be used as rows and which ones as columns.
colour	Columns to use to determine the colors.

Value

A ggplot2 object.

`filterAdditional` *Filter the additional_name-additional_level pair in a summarised_result*

Description

Filter the additional_name-additional_level pair in a summarised_result

Usage

```
filterAdditional(result, ...)
```

Arguments

`result`	A <summarised_result> object.
`...`	Expressions that return a logical value (`additionalColumns()` are used to evaluate the expression), and are defined in terms of the variables in .data. If multiple expressions are included, they are combined with the & operator. Only rows for which all conditions evaluate to TRUE are kept.

Value

A <summarised_result> object with only the rows that fulfill the required specified additional.

Examples

```
library(dplyr)
library(omopgenerics)

x <- tibble(
  "result_id" = 1L,
  "cdm_name" = "eunomia",
  "group_name" = "cohort_name",
  "group_level" = c("cohort1", "cohort2", "cohort3"),
  "strata_name" = "sex",
  "strata_level" = "Female",
  "variable_name" = "number subjects",
  "variable_level" = NA_character_,
  "estimate_name" = "count",
  "estimate_type" = "integer",
  "estimate_value" = c("100", "44", "14"),
  "additional_name" = c("year", "time_step", "year &&& time_step"),
  "additional_level" = c("2010", "4", "2015 &&& 5")
) |>
  newSummarisedResult()

x |>
  filterAdditional(year == "2010")
```

`filterGroup` *Filter the group_name-group_level pair in a summarised_result*

Description

Filter the group_name-group_level pair in a summarised_result

Usage

```
filterGroup(result, ...)
```

Arguments

`result`	A <summarised_result> object.
`...`	Expressions that return a logical value (`groupColumns()` are used to evaluate the expression), and are defined in terms of the variables in .data. If multiple expressions are included, they are combined with the & operator. Only rows for which all conditions evaluate to TRUE are kept.

Value

A <summarised_result> object with only the rows that fulfill the required specified group.

Examples

```
library(dplyr)
library(omopgenerics)

x <- tibble(
  "result_id" = 1L,
  "cdm_name" = "eunomia",
  "group_name" = c("cohort_name", "age_group &&& cohort_name", "age_group"),
  "group_level" = c("my_cohort", ">40 &&& second_cohort", "<40"),
  "strata_name" = "sex",
  "strata_level" = "Female",
  "variable_name" = "number subjects",
  "variable_level" = NA_character_,
  "estimate_name" = "count",
  "estimate_type" = "integer",
  "estimate_value" = c("100", "44", "14"),
  "additional_name" = "overall",
  "additional_level" = "overall"
) |>
  newSummarisedResult()

x |>
  filterGroup(cohort_name == "second_cohort")
```

filterSettings *Filter a* <summarised_result> *using the settings*

Description

Filter a <summarised_result> using the settings

Usage

```
filterSettings(result, ...)
```

Arguments

result	A <summarised_result> object.
...	Expressions that return a logical value (columns in settings are used to evaluate the expression), and are defined in terms of the variables in .data. If multiple expressions are included, they are combined with the & operator. Only rows for which all conditions evaluate to TRUE are kept.

Value

A <summarised_result> object with only the result_id rows that fulfill the required specified settings.

Examples

```
library(dplyr)
library(omopgenerics)

x <- tibble(
  "result_id" = as.integer(c(1, 2)),
  "cdm_name" = c("cprd", "eunomia"),
  "group_name" = "sex",
  "group_level" = "male",
  "strata_name" = "sex",
  "strata_level" = "male",
  "variable_name" = "Age group",
  "variable_level" = "10 to 50",
  "estimate_name" = "count",
  "estimate_type" = "numeric",
  "estimate_value" = "5",
  "additional_name" = "overall",
  "additional_level" = "overall"
) |>
  newSummarisedResult(settings = tibble(
    "result_id" = c(1, 2), "custom" = c("A", "B")
  ))

x
```

```
x |> filterSettings(custom == "A")
```

`filterStrata` *Filter the strata_name-strata_level pair in a summarised_result*

Description

Filter the strata_name-strata_level pair in a summarised_result

Usage

```
filterStrata(result, ...)
```

Arguments

`result`	A <summarised_result> object.
`...`	Expressions that return a logical value (`strataColumns()` are used to evaluate the expression), and are defined in terms of the variables in .data. If multiple expressions are included, they are combined with the & operator. Only rows for which all conditions evaluate to TRUE are kept.

Value

A <summarised_result> object with only the rows that fulfill the required specified strata.

Examples

```
library(dplyr)
library(omopgenerics)

x <- tibble(
  "result_id" = 1L,
  "cdm_name" = "eunomia",
  "group_name" = "cohort_name",
  "group_level" = "my_cohort",
  "strata_name" = c("sex", "sex &&& age_group", "sex &&& year"),
  "strata_level" = c("Female", "Male &&& <40", "Female &&& 2010"),
  "variable_name" = "number subjects",
  "variable_level" = NA_character_,
  "estimate_name" = "count",
  "estimate_type" = "integer",
  "estimate_value" = c("100", "44", "14"),
  "additional_name" = "overall",
  "additional_level" = "overall"
) |>
  newSummarisedResult()
```

```
x |>
  filterStrata(sex == "Female")
```

`formatEstimateName` *Formats estimate_name and estimate_value column*

Description

Formats estimate_name and estimate_value columns by changing the name of the estimate name and/or joining different estimates together in a single row.

Usage

```
formatEstimateName(
  result,
  estimateName = NULL,
  keepNotFormatted = TRUE,
  useFormatOrder = TRUE,
  estimateNameFormat = lifecycle::deprecated()
)
```

Arguments

`result`	A <summarised_result>.
`estimateName`	Named list of estimate name's to join, sorted by computation order. Indicate estimate_name's between <...>.
`keepNotFormatted`	Whether to keep rows not formatted.
`useFormatOrder`	Whether to use the order in which estimate names appear in the estimateName (TRUE), or use the order in the input dataframe (FALSE).
`estimateNameFormat`	deprecated.

Value

A <summarised_result> object.

Examples

```
result <- mockSummarisedResult()
result |>
  formatEstimateName(
    estimateName = c(
      "N (%)" = "<count> (<percentage>%)", "N" = "<count>"
    ),
    keepNotFormatted = FALSE
  )
```

formatEstimateValue *Formats the estimate_value column*

Description

Formats the estimate_value column of <summarised_result> object by editing number of decimals, decimal and thousand/millions separator marks.

Usage

```
formatEstimateValue(
  result,
  decimals = c(integer = 0, numeric = 2, percentage = 1, proportion = 3),
  decimalMark = ".",
  bigMark = ","
)
```

Arguments

result	A <summarised_result>.
decimals	Number of decimals per estimate type (integer, numeric, percentage, proportion), estimate name, or all estimate values (introduce the number of decimals).
decimalMark	Decimal separator mark.
bigMark	Thousand and millions separator mark.

Value

A <summarised_result>.

Examples

```
result <- mockSummarisedResult()

result |> formatEstimateValue(decimals = 1)

result |> formatEstimateValue(decimals = c(integer = 0, numeric = 1))

result |>
  formatEstimateValue(decimals = c(numeric = 1, count = 0))
```

formatHeader *Create a header for gt and flextable objects*

Description

Pivots a <summarised_result> object based on the column names in header, generating specific column names for subsequent header formatting in formatTable function.

Usage

```
formatHeader(
  result,
  header,
  delim = "\n",
  includeHeaderName = TRUE,
  includeHeaderKey = TRUE
)
```

Arguments

result	A <summarised_result>.
header	Names of the variables to make headers.
delim	Delimiter to use to separate headers.
includeHeaderName	Whether to include the column name as header.
includeHeaderKey	Whether to include the header key (header, header_name, header_level) before each header type in the column names.

Value

A tibble with rows pivotted into columns with key names for subsequent header formatting.

Examples

```
result <- mockSummarisedResult()

result |>
  formatHeader(
    header = c(
      "Study cohorts", "group_level", "Study strata", "strata_name",
      "strata_level"
    ),
    includeHeaderName = FALSE
  )
```

formatTable *Creates a flextable or gt object from a dataframe*

Description

Creates a flextable object from a dataframe using a delimiter to span the header, and allows to easily customise table style.

Usage

```
formatTable(
  x,
  type = "gt",
  delim = "\n",
  style = "default",
  na = "-",
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  groupColumn = NULL,
  groupAsColumn = FALSE,
  groupOrder = NULL,
  merge = NULL
)
```

Arguments

x	A dataframe.
type	The desired format of the output table. See `tableType()` for allowed options. If "tibble", no formatting will be applied.
delim	Delimiter.
style	Named list that specifies how to style the different parts of the gt or flextable table generated. Accepted style entries are: title, subtitle, header, header_name, header_level, column_name, group_label, and body. Alternatively, use "default" to get visOmopResults style, or NULL for gt/flextable style. Keep in mind that styling code is different for gt and flextable. To see the "deafult" style code use `tableStyle()`.
na	How to display missing values.
title	Title of the table, or NULL for no title.
subtitle	Subtitle of the table, or NULL for no subtitle.
caption	Caption for the table, or NULL for no caption. Text in markdown formatting style (e.g. `*Your caption here*` for caption in italics).
groupColumn	Specifies the columns to use for group labels. By default, the new group name will be a combination of the column names, joined by "_". To assign a custom group name, provide a named list such as: list(`newGroupName` = c("variable_name", "variable_level"))

groupAsColumn	Whether to display the group labels as a column (TRUE) or rows (FALSE).
groupOrder	Order in which to display group labels.
merge	Names of the columns to merge vertically when consecutive row cells have identical values. Alternatively, use "all_columns" to apply this merging to all columns, or use NULL to indicate no merging.

Value

A flextable object.

A flextable or gt object.

Examples

```
# Example 1
mockSummarisedResult() |>
  formatEstimateValue(decimals = c(integer = 0, numeric = 1)) |>
  formatHeader(
    header = c("Study strata", "strata_name", "strata_level"),
    includeHeaderName = FALSE
  ) |>
  formatTable(
    type = "flextable",
    style = "default",
    na = "--",
    title = "fxTable example",
    subtitle = NULL,
    caption = NULL,
    groupColumn = "group_level",
    groupAsColumn = TRUE,
    groupOrder = c("cohort1", "cohort2"),
    merge = "all_columns"
  )

# Example 2
mockSummarisedResult() |>
  formatEstimateValue(decimals = c(integer = 0, numeric = 1)) |>
  formatHeader(header = c("Study strata", "strata_name", "strata_level"),
               includeHeaderName = FALSE) |>
  formatTable(
    type = "gt",
    style = list("header" = list(
      gt::cell_fill(color = "#d9d9d9"),
      gt::cell_text(weight = "bold")),
      "header_level" = list(gt::cell_fill(color = "#e1e1e1"),
                            gt::cell_text(weight = "bold")),
      "column_name" = list(gt::cell_text(weight = "bold")),
      "title" = list(gt::cell_text(weight = "bold"),
                     gt::cell_fill(color = "#c8c8c8")),
      "group_label" = gt::cell_fill(color = "#e1e1e1")),
    na = "--",
    title = "gtTable example",
```

```
  subtitle = NULL,
  caption = NULL,
  groupColumn = "group_level",
  groupAsColumn = FALSE,
  groupOrder = c("cohort1", "cohort2"),
  merge = "all_columns"
)
```

`fxTable` *Creates a flextable object from a dataframe*

Description

[Deprecated] Creates a flextable object from a dataframe using a delimiter to span the header, and allows to easily customise table style.

Usage

```
fxTable(
  x,
  delim = "\n",
  style = "default",
  na = "-",
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  groupColumn = NULL,
  groupAsColumn = FALSE,
  groupOrder = NULL,
  colsToMergeRows = NULL
)
```

Arguments

`x`	A dataframe.
`delim`	Delimiter.
`style`	Named list that specifies how to style the different parts of the gt or flextable table generated. Accepted style entries are: title, subtitle, header, header_name, header_level, column_name, group_label, and body. Alternatively, use "default" to get visOmopResults style, or NULL for gt/flextable style. Keep in mind that styling code is different for gt and flextable. To see the "deafult" gt style code use `tableStyle()`.
`na`	How to display missing values.
`title`	Title of the table, or NULL for no title.
`subtitle`	Subtitle of the table, or NULL for no subtitle.

caption	Caption for the table, or NULL for no caption. Text in markdown formatting style (e.g. *Your caption here* for caption in italics).
groupColumn	Specifies the columns to use for group labels. By default, the new group name will be a combination of the column names, joined by "_". To assign a custom group name, provide a named list such as: list(newGroupName = c("variable_name", "variable_level"))
groupAsColumn	Whether to display the group labels as a column (TRUE) or rows (FALSE).
groupOrder	Order in which to display group labels.
colsToMergeRows	Names of the columns to merge vertically when consecutive row cells have identical values. Alternatively, use "all_columns" to apply this merging to all columns, or use NULL to indicate no merging.

Value

A flextable object.

A flextable object.

---

groupColumns *Identify variables in group_name column*

Description

Identifies and returns the unique values in group_name column.

Usage

```
groupColumns(result)
```

Arguments

result	A tibble.

Value

Unique values of the group name column.

Examples

```
mockSummarisedResult() |>
  groupColumns()
```

gtTable *Creates a gt object from a dataframe*

Description

[Deprecated] Creates a flextable object from a dataframe using a delimiter to span the header, and allows to easily customise table style.

Usage

```
gtTable(
  x,
  delim = "\n",
  style = "default",
  na = "-",
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  groupColumn = NULL,
  groupAsColumn = FALSE,
  groupOrder = NULL,
  colsToMergeRows = NULL
)
```

Arguments

x	A dataframe.
delim	Delimiter.
style	Named list that specifies how to style the different parts of the gt or flextable table generated. Accepted style entries are: title, subtitle, header, header_name, header_level, column_name, group_label, and body. Alternatively, use "default" to get visOmopResults style, or NULL for gt/flextable style. Keep in mind that styling code is different for gt and flextable. To see the "deafult" style code use `tableStyle()`.
na	How to display missing values.
title	Title of the table, or NULL for no title.
subtitle	Subtitle of the table, or NULL for no subtitle.
caption	Caption for the table, or NULL for no caption. Text in markdown formatting style (e.g. `*Your caption here*` for caption in italics).
groupColumn	Specifies the columns to use for group labels. By default, the new group name will be a combination of the column names, joined by "_". To assign a custom group name, provide a named list such as: list(`newGroupName` = c("variable_name", "variable_level"))
groupAsColumn	Whether to display the group labels as a column (TRUE) or rows (FALSE).

groupOrder Order in which to display group labels.

colsToMergeRows
Names of the columns to merge vertically when consecutive row cells have identical values. Alternatively, use "all_columns" to apply this merging to all columns, or use NULL to indicate no merging.

Value

gt object.

A gt table.

---

mockSummarisedResult *A <summarised_result> object filled with mock data*

Description

Creates an object of the class <summarised_result> with mock data for illustration purposes.

Usage

```
mockSummarisedResult()
```

Value

An object of the class <summarised_result> with mock data.

Examples

```
mockSummarisedResult()
```

---

optionsVisOmopTable *Additional table formatting options*

Description

[Deprecated]

Usage

```
optionsVisOmopTable()
```

Value

list of options

pivotEstimates *Set estimates as columns*

Description

Pivot the estimates as new columns in result table.

Usage

```
pivotEstimates(result, pivotEstimatesBy = "estimate_name", nameStyle = NULL)
```

Arguments

result	A <summarised_result>.
pivotEstimatesBy	Names from which pivot wider the estimate values. If NULL the table will not be pivotted.
nameStyle	Name style (glue package specifications) to customise names when pivotting estimates. If NULL standard tidyr::pivot_wider formatting will be used.

Value

A tibble.

Examples

```
result <- mockSummarisedResult()
result |> pivotEstimates()
```

scatterPlot *Create a scatter plot visualisation from a* <summarised_result> *object*

Description

[Experimental]

Usage

```
scatterPlot(
  result,
  x,
  y,
  line,
  point,
  ribbon,
  ymin = NULL,
  ymax = NULL,
  facet = NULL,
  colour = NULL,
  group = colour
)
```

Arguments

result	A <summarised_result> object.
x	Column or estimate name that is used as x variable.
y	Column or estimate name that is used as y variable
line	Whether to plot a line using geom_line.
point	Whether to plot points using geom_point.
ribbon	Whether to plot a ribbon using geom_ribbon.
ymin	Lower limit of error bars, if provided is plot using geom_errorbar.
ymax	Upper limit of error bars, if provided is plot using geom_errorbar.
facet	Variables to facet by, a formula can be provided to specify which variables should be used as rows and which ones as columns.
colour	Columns to use to determine the colors.
group	Columns to use to determine the group.

Value

A plot object.

Examples

```
result <- mockSummarisedResult() |>
  dplyr::filter(variable_name == "age")

scatterPlot(
  result = result,
  x = "cohort_name",
  y = "mean",
  line = TRUE,
  point = TRUE,
  ribbon = FALSE,
```

```
  facet = age_group ~ sex)
```

`settingsColumns` *Identify settings columns of a* `<summarised_result>`

Description

Identifies and returns the columns of the settings table obtained by using `settings()` in a `<summarised_result>` object.

Usage

```
settingsColumns(result)
```

Arguments

`result`	A `<summarised_result>`.

Value

Vector with names of the settings columns

Examples

```
mockSummarisedResult() |>
  settingsColumns()
```

`splitAdditional` *Split additional_name and additional_level columns*

Description

Pivots the input dataframe so the values of the column additional_name are transformed into columns that contain values from the additional_level column.

Usage

```
splitAdditional(result, keep = FALSE, fill = "overall")
```

Arguments

`result`	A dataframe with at least the columns additional_name and additional_level.
`keep`	Whether to keep the original group_name and group_level columns.
`fill`	Optionally, a character that specifies what value should be filled in with when missing.

Value

A dataframe.

Examples

```
mockSummarisedResult() |>
  splitAdditional()
```

`splitAll` *Split all pairs name-level into columns.*

Description

Pivots the input dataframe so any pair name-level columns are transformed into columns (name) that contain values from the corresponding level.

Usage

```
splitAll(result, keep = FALSE, fill = "overall", exclude = "variable")
```

Arguments

`result`	A data.frame.
`keep`	Whether to keep the original name-level columns.
`fill`	A character that specifies what value should be filled in when missing.
`exclude`	Name of a column pair to exclude.

Value

A dataframe with group, strata and additional as columns.

Examples

```
mockSummarisedResult() |>
  splitAll()
```

splitGroup *Split group_name and group_level columns*

Description

Pivots the input dataframe so the values of the column group_name are transformed into columns that contain values from the group_level column.

Usage

```
splitGroup(result, keep = FALSE, fill = "overall")
```

Arguments

result	A dataframe with at least the columns group_name and group_level.
keep	Whether to keep the original group_name and group_level columns.
fill	Optionally, a character that specifies what value should be filled in with when missing.

Value

A dataframe.

Examples

```
mockSummarisedResult() |>
  splitGroup()
```

splitNameLevel *Split name and level columns into the columns*

Description

[Deprecated] Pivots the input dataframe so the values of the name columns are transformed into columns, which values come from the specified level column.

Usage

```
splitNameLevel(
  result,
  name = "group_name",
  level = "group_level",
  keep = FALSE,
  fill = "overall"
)
```

Arguments

result	A <summarised_result> object.
name	Column with the names.
level	Column with the levels.
keep	Whether to keep the original group_name and group_level columns.
fill	Optionally, a character that specifies what value should be filled in with when missing.

Value

A dataframe with the specified name column values as columns.

---

splitStrata *Split strata_name and strata_level columns*

Description

Pivots the input dataframe so the values of the column strata_name are transformed into columns that contain values from the strata_level column.

Usage

```
splitStrata(result, keep = FALSE, fill = "overall")
```

Arguments

result	A dataframe with at least the columns strata_name and strata_level.
keep	Whether to keep the original group_name and group_level columns.
fill	Optionally, a character that specifies what value should be filled in with when missing.

Value

A dataframe.

Examples

```
mockSummarisedResult() |>
  splitStrata()
```

strataColumns *Identify variables in strata_name column*

Description

Identifies and returns the unique values in strata_name column.

Usage

```
strataColumns(result)
```

Arguments

result	A tibble.

Value

Unique values of the strata name column.

Examples

```
mockSummarisedResult() |>
  strataColumns()
```

tableOptions *Additional table formatting options for* visOmopTable() *and* visTable()

Description

This function provides a list of allowed inputs for the .option argument in visOmopTable() and visTable(), and their corresponding default values.

Usage

```
tableOptions()
```

Value

A named list of default options for table customization.

Examples

```
tableOptions()
```

tableStyle *Supported predefined styles for formatted tables*

Description

Supported predefined styles for formatted tables

Usage

```
tableStyle(type = "gt", styleName = "default")
```

Arguments

type	Character string specifying the formatted table class. See tableType() for supported classes. Default is "gt".
styleName	A character string specifying the style name. Currently, the package supports only one predefined style: "default".

Value

A code expression for the selected style and table type.

Examples

```
tableStyle("gt")
tableStyle("flextable")
```

tableType *Supported table classes*

Description

This function returns the supported table classes that can be used in the type argument of visOmopTable(), visTable(), and formatTable() functions.

Usage

```
tableType()
```

Value

A character vector of supported table types.

Examples

```
tableType()
```

tidy.summarised_result *Turn a* <summarised_result> *object into a tidy tibble*

Description

[Experimental] Provides tools for obtaining a tidy version of a <summarised_result> object. This tidy version will include the settings as columns, estimate_value will be pivotted into columns using estimate_name as names, and group, strata, and additional will be splitted. If you want to customise these tidy operations, please use tidySummarisedResult().

Usage

```
## S3 method for class 'summarised_result'
tidy(x, ...)
```

Arguments

x	A <summarised_result>.
...	For compatibility (not used).

Value

A tibble.

Examples

```
result <- mockSummarisedResult()
result |> tidy()
```

tidyColumns *Identify tidy columns of a* <summarised_result>

Description

Identifies and returns the columns that the tidy version of the <summarised_result> will have.

Usage

```
tidyColumns(result)
```

Arguments

result	A <summarised_result>.

Value

Table columns after applying `tidy()` function to a `<summarised_result>`.

Examples

```
mockSummarisedResult() |>
  tidyColumns()
```

`uniteAdditional` *Unite one or more columns in additional_name-additional_level format*

Description

Unites targeted table columns into additional_name-additional_level columns.

Usage

```
uniteAdditional(
  x,
  cols = character(0),
  keep = FALSE,
  ignore = c(NA, "overall")
)
```

Arguments

`x`	Tibble or dataframe.
`cols`	Columns to aggregate.
`keep`	Whether to keep the original columns.
`ignore`	Level values to ignore.

Value

A tibble with the new columns.

Examples

```
x <- dplyr::tibble(
  variable = "number subjects",
  value = c(10, 15, 40, 78),
  sex = c("Male", "Female", "Male", "Female"),
  age_group = c("<40", ">40", ">40", "<40")
)

x |>
  uniteAdditional(c("sex", "age_group"))
```

uniteGroup *Unite one or more columns in group_name-group_level format*

Description

Unites targeted table columns into group_name-group_level columns.

Usage

```
uniteGroup(x, cols = character(0), keep = FALSE, ignore = c(NA, "overall"))
```

Arguments

x	Tibble or dataframe.
cols	Columns to aggregate.
keep	Whether to keep the original columns.
ignore	Level values to ignore.

Value

A tibble with the new columns.

Examples

```
x <- dplyr::tibble(
  variable = "number subjects",
  value = c(10, 15, 40, 78),
  sex = c("Male", "Female", "Male", "Female"),
  age_group = c("<40", ">40", ">40", "<40")
)

x |>
  uniteGroup(c("sex", "age_group"))
```

uniteNameLevel *Unite one or more columns in name-level format*

Description

[Deprecated] Unites targeted table columns into a pair of name-level columns.

Usage

```
uniteNameLevel(
  x,
  cols = character(0),
  name = "group_name",
  level = "group_level",
  keep = FALSE,
  ignore = c(NA, "overall")
)
```

Arguments

x	A dataframe.
cols	Columns to aggregate.
name	Column name of the name column.
level	Column name of the level column.
keep	Whether to keep the original columns.
ignore	Level values to ignore.

Value

A tibble with the new columns.

---

uniteStrata *Unite one or more columns in strata_name-strata_level format*

Description

Unites targeted table columns into strata_name-strata_level columns.

Usage

```
uniteStrata(x, cols = character(0), keep = FALSE, ignore = c(NA, "overall"))
```

Arguments

x	Tibble or dataframe.
cols	Columns to aggregate.
keep	Whether to keep the original columns.
ignore	Level values to ignore.

Value

A tibble with the new columns.

Examples

```
x <- dplyr::tibble(
  variable = "number subjects",
  value = c(10, 15, 40, 78),
  sex = c("Male", "Female", "Male", "Female"),
  age_group = c("<40", ">40", ">40", "<40")
)

x |>
  uniteStrata(c("sex", "age_group"))
```

---

visOmopTable *Format a* <summarised_result> *object into a gt, flextable, or tibble object*

Description

This function combines the functionalities of `formatEstimateValue()`, `estimateName()`, `formatHeader()`, and `formatTable()` into a single function specifically for <summarised_result> objects.

Usage

```
visOmopTable(
  result,
  estimateName = character(),
  header = character(),
  settingsColumns = character(),
  groupColumn = character(),
  rename = character(),
  type = "gt",
  hide = character(),
  showMinCellCount = TRUE,
  .options = list(),
  split = lifecycle::deprecated(),
  excludeColumns = lifecycle::deprecated(),
  formatEstimateName = lifecycle::deprecated(),
  renameColumns = lifecycle::deprecated()
)
```

Arguments

result	A <summarised_result> object.
estimateName	A named list of estimate names to join, sorted by computation order. Use <...> to indicate estimate names.
header	A vector specifying the elements to include in the header. The order of elements matters, with the first being the topmost header. The input vector elements can be:

1. Column names from the split summarised result generated by `splitAll()`
2. Settings specified in the `settings` argument
3. `group`, `strata`, `additional`, `variable`, `estimate`, and/or `settings` to refer to all columns within these groups
4. Any other input to create overall header labels at the specified location.

`settingsColumns`
: A character vector with the names of settings to include in the table.

`groupColumn`
: Columns to use as group labels. By default, the name of the new group will be the tidy* column names separated by ";". To specify a custom group name, use a named list such as: list("newGroupName" = c("variable_name", "variable_level")).
*tidy: The tidy format applied to column names replaces "_" with a space and converts to sentence case. Use `rename` to customize specific column names.

`rename`
: A named vector to customize column names, e.g., c("Database name" = "cdm_name"). The function renames all column names not specified here into a tidy* format.

`type`
: The desired format of the output table. See `tableType()` for allowed options.

`hide`
: Columns to drop from the output table. By default, `result_id` and `estimate_type` are always dropped.

`showMinCellCount`
: If `TRUE`, suppressed estimates will be indicated with "<{min_cell_count}", otherwise, the default `na` defined in `.options` will be used.

`.options`
: A named list with additional formatting options. `visOmopResults::tableOptions()` shows allowed arguments and their default values.

`split`
: **[Deprecated]**

`excludeColumns`
: **[Deprecated]**

`formatEstimateName`
: **[Deprecated]**

`renameColumns`
: **[Deprecated]**

Value

A tibble, gt, or flextable object.

Examples

```
result <- mockSummarisedResult()
result |>
  visOmopTable(
    estimateName = c("N%" = "<count> (<percentage>)",
                     "N" = "<count>",
                     "Mean (SD)" = "<mean> (<sd>)"),
    header = c("group"),
    rename = c("Database name" = "cdm_name"),
    groupColumn = strataColumns(result)
  )
```

visTable *Generate a formatted table from a* <data.table>

Description

[Experimental] This function combines the functionalities of formatEstimateValue(), formatEstimateName(), formatHeader(), and formatTable() into a single function. While it does not require the input table to be a <summarised_result>, it does expect specific fields to apply some formatting functionalities.

Usage

```
visTable(
  result,
  estimateName = character(),
  header = character(),
  groupColumn = character(),
  rename = character(),
  type = "gt",
  hide = character(),
  .options = list()
)
```

Arguments

result	A table to format.
estimateName	A named list of estimate names to join, sorted by computation order. Use <...> to indicate estimate names. This argument requires that the table has estimate_name and estimate_value columns.
header	A vector specifying the elements to include in the header. The order of elements matters, with the first being the topmost header. The vector elements can be column names or labels for overall headers. The table must contain an estimate_value column to pivot the headers.
groupColumn	Columns to use as group labels. By default, the name of the new group will be the tidy* column names separated by ";". To specify a custom group name, use a named list such as: list("newGroupName" = c("variable_name", "variable_level")). *tidy: The tidy format applied to column names replaces "_" with a space and converts them to sentence case. Use rename to customize specific column names.
rename	A named vector to customize column names, e.g., c("Database name" = "cdm_name"). The function will rename all column names not specified here into a tidy* format.
type	The desired format of the output table. See tableType() for allowed options.
hide	Columns to drop from the output table.

`.options` A named list with additional formatting options. `visOmopResults::tableOptions()` shows allowed arguments and their default values.

Value

A tibble, gt, or flextable object.

Examples

```
result <- mockSummarisedResult()
result |>
  visTable(
    estimateName = c("N%" = "<count> (<percentage>)",
                     "N" = "<count>",
                     "Mean (SD)" = "<mean> (<sd>)"),
    header = c("Estimate"),
    rename = c("Database name" = "cdm_name"),
    groupColumn = c("strata_name", "strata_level"),
    hide = c("additional_name", "additional_level", "estimate_type", "result_type")
  )
```

Index