Package 'visOmopResults'

May 18, 2026

Title Graphs and Tables for OMOP Results

Version 1.5.0

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 visualisation 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/, https://github.com/darwin-eu/visOmopResults

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

Imports brand.yml, cli, dplyr, generics, glue, omopgenerics (>= 0.3.1), purrr, rlang, stringr, systemfonts, tidyr

Suggests bslib, CohortCharacteristics, covr, DT, flextable (>= 0.9.5), ggalluvial, ggplot2, ggsankeyfier, gt, here, htmltools, IncidencePrevalence, knitr, lifecycle, officer, palmerpenguins, PatientProfiles, plotly, reactable, rmarkdown, shiny, shinycssloaders, shinyWidgets, sortable, testthat (>= 3.0.0), tinytable (>= 0.14.0), yaml

VignetteBuilder knitr

Depends R (>= 4.1)

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

LazyData true

Config/roxygen2/version 8.0.0

NeedsCompilation no

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

Yuchen Guo [ctb] (ORCID: <https://orcid.org/0000-0002-0847-4855>),
Elin Rowlands [ctb] (ORCID: <https://orcid.org/0009-0007-6629-4661>),
Marta Alcalde-Herraiz [ctb] (ORCID:
<https://orcid.org/0009-0002-4405-1814>),
Edward Burn [ctb] (ORCID: <https://orcid.org/0000-0002-9286-1128>)

Repository CRAN

Date/Publication 2026-05-18 05:20:02 UTC

Contents

`alluvialPlot` *Create an alluvial plot visualisation from a data frame or a* `<summarised_result>` *object.*

Description

Create an alluvial plot visualisation from a data frame or a `<summarised_result>` object.

Usage

```
alluvialPlot(result, x, y, colour = x, facet = NULL, style = NULL, type = NULL)
```

Arguments

result	A <summarised_result> object.
x	A character vector of column names to use as alluvial axes, in order from left to right. Must contain at least 2 elements.
y	Column or estimate name that is used as y variable.
colour	Columns to use to determine the colours.
facet	Variables to facet by, a formula can be provided to specify which variables should be used as rows and which ones as columns.
style	Visual theme to apply. Character, or NULL. If a character, this may be either the name of a built-in style (see plotStyle()), or a path to a .yml file that defines a custom style. If NULL, the function will use the explicit default style, unless a global style option is set (see setGlobalPlotOptions()), or a _brand.yml file is present (in that order). Refer to the package vignette on styles to learn more.
type	Character string indicating the output plot format. See plotType() for the list of supported plot types. If type = NULL, the function will use the global setting defined via setGlobalPlotOptions() (if available); otherwise, a standard ggplot2 plot is produced by default.

Value

A plot object.

Examples

```
result <- dplyr::tibble(
  treatment_1 = c("A", "A", "A", "B", "B", "B", "C", "C"),
  treatment_2 = c("A", "A", "B", "A", "B", "B", "B", "C"),
  treatment_3 = c("A", "B", "B", "A", "A", "B", "B", "C"),
  count       = c(22, 3, 5, 7, 3, 17, 4, 12)
)

# basic alluvial plot with 3 axes
alluvialPlot(
  result = result,
  x = c("treatment_1", "treatment_2", "treatment_3"),
  y = "count"
)

# colour by first axis
alluvialPlot(
  result = result,
  x = c("treatment_1", "treatment_2", "treatment_3"),
  y = "count",
  colour = "treatment_1"
```

```
)

# colour by multiple variables
alluvialPlot(
  result = result,
  x = c("treatment_1", "treatment_2", "treatment_3"),
  y = "count",
  colour = c("treatment_1", "treatment_2")
)
```

---

barPlot *Create a bar plot visualisation from a data frame or a* <summarised_result> *object.*

Description

Create a bar plot visualisation from a data frame or a <summarised_result> object.

Usage

```
barPlot(
  result,
  x,
  y,
  width = NULL,
  just = 0.5,
  position = "dodge",
  facet = NULL,
  colour = NULL,
  style = NULL,
  type = NULL,
  label = character()
)
```

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.
width	Bar width, as in geom_col() of the ggplot2 package.
just	Adjustment for column placement, as in geom_col() of the ggplot2 package.
position	Position of bars, can be either dodge or stack
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 colours.

style	Visual theme to apply. Character, or NULL. If a character, this may be either the name of a built-in style (see plotStyle()), or a path to a .yml file that defines a custom style. If NULL, the function will use the explicit default style, unless a global style option is set (see setGlobalPlotOptions()), or a _brand.yml file is present (in that order). Refer to the package vignette on styles to learn more.
type	Character string indicating the output plot format. See plotType() for the list of supported plot types. If type = NULL, the function will use the global setting defined via setGlobalPlotOptions() (if available); otherwise, a standard ggplot2 plot is produced by default.
label	Character vector with the columns to display interactively in plotly.

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 data frame or a <summarised_result> object.*

Description

Create a box plot visualisation from a data frame or a <summarised_result> object.

Usage

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

```
  style = NULL,
  type = NULL,
  label = character()
)
```

Arguments

result	A <summarised_result> object.
x	Column or estimate name that is used as x variable.
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	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 colours.
style	Visual theme to apply. Character, or NULL. If a character, this may be either the name of a built-in style (see plotStyle()), or a path to a .yml file that defines a custom style. If NULL, the function will use the explicit default style, unless a global style option is set (see setGlobalPlotOptions()), or a _brand.yml file is present (in that order). Refer to the package vignette on styles to learn more.
type	Character string indicating the output plot format. See plotType() for the list of supported plot types. If type = NULL, the function will use the global setting defined via setGlobalPlotOptions() (if available); otherwise, a standard ggplot2 plot is produced by default.
label	Character vector with the columns to display interactively in plotly.

Value

A ggplot2 object.

Examples

```
dplyr::tibble(year = "2000", q25 = 25, median = 50, q75 = 75, min = 0, max = 100) |>
  boxPlot(x = "year")
```

customiseText *Apply styling to text or column names*

Description

This function styles character vectors or column names in a data frame. The styling function can be customised, or you can provide specific replacements for certain values.

Usage

```
customiseText(
  x,
  fun = function(x) stringr::str_to_sentence(gsub("_", " ", x)),
  custom = NULL,
  keep = NULL
)
```

Arguments

x	A character vector to style text.
fun	A styling function to apply to text in x. The default function converts snake_case to sentence case.
custom	A named character vector indicating custom names for specific values in x. If NULL, the styling function in fun is applied to all values.
keep	Either a character vector of names to keep unchanged. If NULL, all names will be styled.

Value

A character vector of styled text or a data frame with styled column names.

Examples

```
# Styling a character vector
customiseText(c("some_column_name", "another_column"))

# Custom styling for specific values
customiseText(x = c("some_column", "another_column"),
          custom = c("Custom Name" = "another_column"))

# Keeping specific values unchanged
customiseText(x = c("some_column", "another_column"), keep = "another_column")

# Styling column names and variables in a data frame
dplyr::tibble(
  some_column = c("hi_there", "rename_me", "example", "to_keep"),
  another_column = 1:4,
  to_keep = "as_is"
```

```
) |>
  dplyr::mutate(
   "some_column" = customiseText(some_column, custom = c("EXAMPLE" = "example"), keep = "to_keep")
  ) |>
  dplyr::rename_with(.fn = ~ customiseText(.x, keep = "to_keep"))
```

`data` *List of mock results*

Description

List of mock results

Usage

```
data
```

Format

A list of mock results for quarto and shiny vignette examples

---

`emptyPlot` *Returns an empty plot*

Description

Returns an empty plot

Usage

```
emptyPlot(title = "No data to plot", subtitle = "", type = NULL, style = NULL)
```

Arguments

`title`	Title to use in the empty plot.
`subtitle`	Subtitle to use in the empty plot.
`type`	Character string indicating the output plot format. See `plotType()` for the list of supported plot types. If `type = NULL`, the function will use the global setting defined via `setGlobalPlotOptions()` (if available); otherwise, a standard `ggplot2` plot is produced by default.
`style`	Visual theme to apply. Character, or `NULL`. If a character, this may be either the name of a built-in style (see `plotStyle()`), or a path to a `.yml` file that defines a custom style. If `NULL`, the function will use the explicit default style, unless a global style option is set (see `setGlobalPlotOptions()`), or a `_brand.yml` file is present (in that order). Refer to the package vignette on styles to learn more.

Value

An empty ggplot object

Examples

```
emptyPlot()
```

`emptyTable` *Returns an empty table*

Description

Returns an empty table

Usage

```
emptyTable(type = NULL, style = NULL)
```

Arguments

`type`	Character string specifying the desired output table format. See `tableType()` for supported table types. If `type = NULL`, global options (set via `setGlobalTableOptions()`) will be used if available; otherwise, a default `'gt'` table is created.
`style`	Defines the visual formatting of the table. This argument can be provided in one of the following ways: 1. **Pre-defined style:** Use the name of a built-in style (e.g., `"darwin"`). See `tableStyle()` for available options. 2. **YAML file path:** Provide the path to an existing `.yml` file defining a new style. 3. **List of custome R code:** Supply a block of custom R code or a named list describing styles for each table section. This code must be specific to the selected table type. If `style = NULL`, the function will use global options (see `setGlobalTableOptions()`) or an existing `_brand.yml` file (if found); otherwise, the default style is applied. For more details, see the *Styles* vignette on the package website.

Value

An empty table of the class specified in `type`

Examples

```
emptyTable(type = "flextable")
```

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
)
```

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).

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
  )
```

formatMinCellCount *To indicate which was the minimum cell counts where estimates have been suppressed.*

Description

To indicate which was the minimum cell counts where estimates have been suppressed.

Usage

```
formatMinCellCount(result)
```

Arguments

result	A <summarised_result> object.

Examples

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

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 = NULL,
  delim = "\n",
  style = NULL,
  na = "-",
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  groupColumn = NULL,
  groupAsColumn = FALSE,
  groupOrder = NULL,
  merge = "all_columns"
)
```

Arguments

x	A dataframe.
type	Character string specifying the desired output table format. See `tableType()` for supported table types. If `type = NULL`, global options (set via `setGlobalTableOptions()`) will be used if available; otherwise, a default `'gt'` table is created.
delim	Delimiter to separate headers.
style	Defines the visual formatting of the table. This argument can be provided in one of the following ways: 1. **Pre-defined style:** Use the name of a built-in style (e.g., `"darwin"`). See `tableStyle()` for available options. 2. **YAML file path:** Provide the path to an existing `.yml` file defining a new style. 3. **List of custome R code:** Supply a block of custom R code or a named list describing styles for each table section. This code must be specific to the selected table type. If `style = NULL`, the function will use global options (see `setGlobalTableOptions()`) or an existing `_brand.yml` file (if found); otherwise, the default style is applied. For more details, see the *Styles* vignette on the package website.
na	How to display missing values. Not used for "datatable" and "reactable".
title	Title of the table, or NULL for no title. Not used for "datatable".
subtitle	Subtitle of the table, or NULL for no subtitle. Not used for "datatable" and "reactable".
caption	Caption for the table, or NULL for no caption. Text in markdown formatting style (e.g. `*Your caption here*` for caption in italics). Not used for "reactable".
groupColumn	Columns to use as group labels, to see options use `tableColumns(result)`. 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 customise specific column names.
groupAsColumn	Whether to display the group labels as a column (TRUE) or rows (FALSE). Not used for "datatable" and "reactable"
groupOrder	Order in which to display group labels. Not used for "datatable" and "reactable".
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. Not used for "datatable" and "reactable".

Value

A formatted table of the class selected in "type" argument.

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"
  )
```

---

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()
```

---

plotColumns *Columns for the plot functions*

Description

Names of the columns that can be used in the input arguments for the plot functions.

Usage

```
plotColumns(result)
```

Arguments

result	A <summarised_result> object.

Value

A character vector of supported columns for plots.

Examples

```
result <- mockSummarisedResult()
plotColumns(result)
```

plotStyle *Pre-defined styles are available for plots*

Description

This function provides a list of pre-defined styles for plots that can then be used in the `style` argument of the plot functions.

Usage

```
plotStyle()
```

Value

A character vector indicating the style names.

Examples

```
plotStyle()
```

plotType *Supported plot types*

Description

This function returns the supported plot types that can be used in the `type` argument of the plot functions.

Usage

```
plotType()
```

Value

A character vector of supported plot types.

Examples

```
tableType()
```

scatterPlot *Create a scatter plot visualisation from a data frame or a* <summarised_result> *object.*

Description

Create a scatter plot visualisation from a data frame or a <summarised_result> object.

Usage

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

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 colours.
style	Visual theme to apply. Character, or NULL. If a character, this may be either the name of a built-in style (see plotStyle()), or a path to a .yml file that defines a custom style. If NULL, the function will use the explicit default style, unless a global style option is set (see setGlobalPlotOptions()), or a _brand.yml file is present (in that order). Refer to the package vignette on styles to learn more.

type	Character string indicating the output plot format. See plotType() for the list of supported plot types. If type = NULL, the function will use the global setting defined via setGlobalPlotOptions() (if available); otherwise, a standard ggplot2 plot is produced by default.
group	Columns to use to determine the group.
label	Character vector with the columns to display interactively in plotly.

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)
```

---

setGlobalPlotOptions *Set format options for all subsequent plots*

Description

Set format options for all subsequent plots unless state a different style in a specific function

Usage

```
setGlobalPlotOptions(style = NULL, type = NULL)
```

Arguments

style	Visual theme to apply. Character, or NULL. If a character, this may be either the name of a built-in style (see plotStyle()), or a path to a .yml file that defines a custom style. If NULL, the function will use the explicit default style, unless a global style option is set (see setGlobalPlotOptions()), or a _brand.yml file is present (in that order). Refer to the package vignette on styles to learn more.
type	Character string indicating the output plot format. See plotType() for the list of supported plot types. If type = NULL, the function will use the global setting defined via setGlobalPlotOptions() (if available); otherwise, a standard ggplot2 plot is produced by default.

`setGlobalTableOptions` *Set format options for all subsequent tables*

Description

Set format options for all subsequent tables unless state a different style in a specific function

Usage

```
setGlobalTableOptions(style = NULL, type = NULL)
```

Arguments

`style`	Defines the visual formatting of the table. This argument can be provided in one of the following ways: 1. **Pre-defined style:** Use the name of a built-in style (e.g., `"darwin"`). See `tableStyle()` for available options. 2. **YAML file path:** Provide the path to an existing `.yml` file defining a new style. 3. **List of custome R code:** Supply a block of custom R code or a named list describing styles for each table section. This code must be specific to the selected table type. If `style = NULL`, the function will use global options (see `setGlobalTableOptions()`) or an existing `_brand.yml` file (if found); otherwise, the default style is applied. For more details, see the *Styles* vignette on the package website.
`type`	Character string specifying the desired output table format. See `tableType()` for supported table types. If `type = NULL`, global options (set via `setGlobalTableOptions()`) will be used if available; otherwise, a default `'gt'` table is created.

Examples

```
setGlobalTableOptions(style = "darwin", type = "tinytable")
result <- mockSummarisedResult()
result |>
  visOmopTable(
    estimateName = c("N%" = "<count> (<percentage>)",
                     "N" = "<count>",
                     "Mean (SD)" = "<mean> (<sd>)"),
    header = c("cohort_name"),
    rename = c("Database name" = "cdm_name"),
    groupColumn = strataColumns(result)
  )
 # drop global options:
 setGlobalTableOptions(style = NULL, type = NULL)
```

tableColumns *Columns for the table functions*

Description

Names of the columns that can be used in the input arguments for the table functions.

Usage

```
tableColumns(result)
```

Arguments

result	A <summarised_result> object.

Value

A character vector of supported columns for tables.

Examples

```
result <- mockSummarisedResult()
tableColumns(result)
```

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 customisation.

Examples

```
tableOptions()
```

tableStyle *Pre-defined styles are available for tables*

Description

This function provides a list of pre-defined styles for tables that can then be used in the `style` argument of the table functions.

Usage

```
tableStyle()
```

Value

A character vector indicating the style names.

Examples

```
tableStyle()
```

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()
```

themeVisOmop *Apply a pre-defined visOmopResults theme to a ggplot*

Description

Apply a pre-defined visOmopResults theme to a ggplot

Usage

```
themeVisOmop(style = NULL, fontsizeRef = NULL)
```

Arguments

style	Visual theme to apply. Character, or NULL. If a character, this may be either the name of a built-in style (see plotStyle()), or a path to a .yml file that defines a custom style. If NULL, the function will use the explicit default style, unless a global style option is set (see setGlobalPlotOptions()), or a _brand.yml file is present (in that order). Refer to the package vignette on styles to learn more.
fontsizeRef	An integer to use as reference when adjusting label fontsize.

Examples

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

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

visOmopTable *Generate a formatted table from a* <summarised_result>

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(),
  settingsColumn = character(),
  groupColumn = character(),
  rename = character(),
  type = NULL,
  hide = character(),
  columnOrder = character(),
  factor = list(),
  style = NULL,
  showMinCellCount = TRUE,
  .options = list()
)
```

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. Elements in header can be: • Any of the columns returned by tableColumns(result) to create a header for these columns. • Any other input to create an overall header.
settingsColumn	A character vector with the names of settings to include in the table. To see options use settingsColumns(result).
groupColumn	Columns to use as group labels, to see options use tableColumns(result). 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 customise specific column names.
rename	A named vector to customise column names, e.g., c("Database name" = "cdm_name"). The function renames all column names not specified here into a tidy* format.
type	Character string specifying the desired output table format. See tableType() for supported table types. If type = NULL, global options (set via setGlobalTableOptions()) will be used if available; otherwise, a default 'gt' table is created.
hide	Columns to drop from the output table. By default, result_id and estimate_type are always dropped.
columnOrder	Character vector establishing the position of the columns in the formatted table. Columns in either header, groupColumn, or hide will be ignored.

`factor` A named list where names refer to columns (see available columns in `tableColumns()`) and list elements are the level order of that column to arrange the results. The column order in the list will be used for arranging the result.

`style` Defines the visual formatting of the table. This argument can be provided in one of the following ways:

1. **Pre-defined style:** Use the name of a built-in style (e.g., `"darwin"`). See `tableStyle()` for available options.
2. **YAML file path:** Provide the path to an existing `.yml` file defining a new style.
3. **List of custome R code:** Supply a block of custom R code or a named list describing styles for each table section. This code must be specific to the selected table type. If `style = NULL`, the function will use global options (see `setGlobalTableOptions()`) or an existing `_brand.yml` file (if found); otherwise, the default style is applied. For more details, see the *Styles* vignette on the package website.

`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.

Value

A formatted table of the class selected in "type" argument.

Examples

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

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

Description

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 = NULL,
  hide = character(),
  style = NULL,
  .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.
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, to see options use tableColumns(result). 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 customise specific column names.
rename	A named vector to customise column names, e.g., c("Database name" = "cdm_name"). The function renames all column names not specified here into a tidy* format.
type	Character string specifying the desired output table format. See tableType() for supported table types. If type = NULL, global options (set via setGlobalTableOptions()) will be used if available; otherwise, a default 'gt' table is created.
hide	Columns to drop from the output table.

`style` Defines the visual formatting of the table. This argument can be provided in one of the following ways:

1. **Pre-defined style:** Use the name of a built-in style (e.g., `"darwin"`). See `tableStyle()` for available options.
2. **YAML file path:** Provide the path to an existing `.yml` file defining a new style.
3. **List of custome R code:** Supply a block of custom R code or a named list describing styles for each table section. This code must be specific to the selected table type. If `style = NULL`, the function will use global options (see `setGlobalTableOptions()`) or an existing `_brand.yml` file (if found); otherwise, the default style is applied. For more details, see the *Styles* vignette on the package website.

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

Value

A formatted table of the class selected in "type" argument.

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