

Package ‘visOmopResults’

October 9, 2025

Title Graphs and Tables for OMOP Results

Version 1.3.1

Maintainer Núria Mercadé-Besora <nuria.mercadebesora@endorms.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 cli, dplyr, generics, glue, omopgenerics (>= 0.3.1), purrr, rlang, stringr, tidyr

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

VignetteBuilder knitr

Depends R (>= 4.1)

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

RoxygenNote 7.3.3

LazyData true

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 2025-10-09 12:20:02 UTC

Contents

barPlot	3
boxPlot	4
customiseText	5
data	6
emptyPlot	7
emptyTable	7
formatEstimateName	8
formatEstimateValue	9
formatHeader	10
formatMinCellCount	11
formatTable	11
mockSummarisedResult	13
plotColumns	14
plotStyle	14
plotType	15
requireExtrafont	15
scatterPlot	16
setGlobalPlotOptions	17
setGlobalTableOptions	18
tableColumns	19
tableOptions	19
tableStyle	20
tableStyleCode	20
tableType	21
themeVisOmop	21
visOmopTable	22
visTable	24

Index	26
--------------	-----------

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

Description

Create a bar plot visualisation from a <summarised_result> object

Usage

```
barPlot(  
  result,  
  x,  
  y,  
  width = NULL,  
  just = 0.5,  
  position = "dodge",  
  facet = NULL,  
  colour = NULL,  
  style = "default",  
  type = "ggplot",  
  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 <code>geom_col()</code> of the <code>ggplot2</code> package.
just	Adjustment for column placement, as in <code>geom_col()</code> of the <code>ggplot2</code> package.
position	Position of bars, can be either <code>dodge</code> or <code>stack</code>
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	A character string defining the visual theme to apply to the plot. You can set this to <code>NULL</code> to apply the standard <code>ggplot2</code> default style, or provide a name for one of the package's pre-defined styles. Refer to the <code>plotStyle()</code> function for all available style pre-defined themes. For further customization, you can always modify the returned <code>ggplot</code> object directly.
type	The desired format of the output plot. See <code>plotType()</code> for supported plot types.
label	Character vector with the columns to display interactively in <code>plotly</code> .

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

Create a box plot visualisation from a <summarised_result> object

Usage

```
boxPlot(
  result,
  x,
  lower = "q25",
  middle = "median",
  upper = "q75",
  ymin = "min",
  ymax = "max",
  facet = NULL,
  colour = NULL,
  style = "default",
  type = "ggplot",
  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	A character string defining the visual theme to apply to the plot. You can set this to NULL to apply the standard ggplot2 default style, or provide a name for one of the package's pre-defined styles. Refer to the plotStyle() function for all available style pre-defined themes. For further customization, you can always modify the returned ggplot object directly.
type	The desired format of the output plot. See plotType() for supported plot types.
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	<i>Apply styling to text or column names</i>
---------------	--

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	<i>List of mock results</i>
------	-----------------------------

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 = "ggplot")
```

Arguments

title	Title to use in the empty plot.
subtitle	Subtitle to use in the empty plot.
type	The desired format of the output plot. See <code>plotType()</code> for supported plot types.

Value

An empty ggplot object

Examples

```
emptyPlot()
```

emptyTable	Returns an empty table
------------	------------------------

Description

Returns an empty table

Usage

```
emptyTable(type = "gt")
```

Arguments

type	The desired format of the output table. See <code>tableType()</code> for allowed options.
------	---

Value

An empty table of the class specified in `type`

Examples

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

formatEstimateName	<i>Formats estimate_name and estimate_value column</i>
--------------------	--

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	<i>Formats the estimate_value column</i>
---------------------	--

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

<code>result</code>	A <summarised_result>.
<code>header</code>	Names of the variables to make headers.
<code>delim</code>	Delimiter to use to separate headers.
<code>includeHeaderName</code>	Whether to include the column name as header.
<code>includeHeaderKey</code>	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	<i>To indicate which was the minimum cell counts where estimates have been suppressed.</i>
--------------------	--

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	<i>Creates a flextable or gt object from a dataframe</i>
-------------	--

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

<code>x</code>	A dataframe.
<code>type</code>	The desired format of the output table. See <code>tableType()</code> for allowed options.
<code>delim</code>	Delimiter to separate headers.
<code>style</code>	A character string or custom R code to define the visual formatting of the table. This argument can be provided in two ways: 1. Pre-defined Styles (Character String): Use a name for a built-in style (e.g., "darwin"). See <code>tableStyle()</code> for available options. 2. Custom Code (Advanced): Supply a block of custom R code. This code must be specific to the table type. See <code>tableStyleCode()</code> for structural examples.
<code>na</code>	How to display missing values. Not used for "datatable" and "reactable".
<code>title</code>	Title of the table, or NULL for no title. Not used for "datatable".
<code>subtitle</code>	Subtitle of the table, or NULL for no subtitle. Not used for "datatable" and "reactable".
<code>caption</code>	Caption for the table, or NULL for no caption. Text in markdown formatting style (e.g. <i>*Your caption here*</i> for caption in italics). Not used for "reactable".
<code>groupColumn</code>	Columns to use as group labels, to see options use <code>tableColumns(result)</code> . 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: <code>list("newGroupName" = c("variable_name", "variable_level"))</code> . *tidy: The tidy format applied to column names replaces "_" with a space and converts to sentence case. Use <code>rename</code> to customise specific column names.
<code>groupAsColumn</code>	Whether to display the group labels as a column (TRUE) or rows (FALSE). Not used for "datatable" and "reactable"
<code>groupOrder</code>	Order in which to display group labels. Not used for "datatable" and "reactable".
<code>merge</code>	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	<i>Supported plot types</i>
----------	-----------------------------

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()
```

requireExtrafont	<i>Get Calibri font for "darwin" plot style</i>
------------------	---

Description

Imports and registers the Calibri font for use in ggplot2 plots when using the themeDarwin.

Usage

```
requireExtrafont(device = "win")
```

Arguments

device	Character string indicating the graphics device to load the fonts into (e.g., "win", "pdf", "postscript"). Defaults to "win" (Windows).
--------	---

Details

This function is required only once per R installation or after installing a new font. It uses the `extrafont` package to map system fonts to R's graphics devices.

Value

Invisibly returns TRUE on successful registration.

scatterPlot	<i>Create a scatter plot visualisation from a <summarised_result> object</i>
-------------	--

Description

Create a scatter plot visualisation from a <summarised_result> object

Usage

```
scatterPlot(
  result,
  x,
  y,
  line,
  point,
  ribbon,
  ymin = NULL,
  ymax = NULL,
  facet = NULL,
  colour = NULL,
  style = "default",
  type = "ggplot",
  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 <code>geom_line</code> .
point	Whether to plot points using <code>geom_point</code> .
ribbon	Whether to plot a ribbon using <code>geom_ribbon</code> .
ymin	Lower limit of error bars, if provided is plot using <code>geom_errorbar</code> .
ymax	Upper limit of error bars, if provided is plot using <code>geom_errorbar</code> .

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	A character string defining the visual theme to apply to the plot. You can set this to NULL to apply the standard ggplot2 default style, or provide a name for one of the package's pre-defined styles. Refer to the plotStyle() function for all available style pre-defined themes. For further customization, you can always modify the returned ggplot object directly.
type	The desired format of the output plot. See plotType() for supported plot types.
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	Character indicating which style to apply. Options are either NULL (default ggplot style), or one of our pre-defined styles (refer to the function plotStyle()).
type	The desired format of the output plot. See plotType() for supported plot types.

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 | A character string or custom R code to define the visual formatting of the table. This argument can be provided in two ways: <ol style="list-style-type: none"> 1. Pre-defined Styles (Character String): Use a name for a built-in style (e.g., "darwin"). See tableStyle() for available options. 2. Custom Code (Advanced): Supply a block of custom R code. This code must be specific to the table type. See tableStyleCode() for structural examples. |
| type | The desired format of the output table. See tableType() for allowed options. |

Examples

```
setGlobalTableOptions(style = "darwin", type = "tinytable")
result <- mockSummarisedResult()
result |>
  visOmapTable(
    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	<i>Columns for the table functions</i>
--------------	--

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	<i>Additional table formatting options for visOmapTable() and visTable()</i>
--------------	--

Description

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

Usage

```
tableOptions()
```

Value

A named list of default options for table customisation.

Examples

```
tableOptions()
```

tableStyle	<i>Pre-defined styles are available for tables</i>
------------	--

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()
```

tableStyleCode	<i>Supported predefined style codes for formatted tables</i>
----------------	--

Description

Supported predefined style codes for formatted tables

Usage

```
tableStyleCode(type = "gt", style = "default")
```

Arguments

- type Character string specifying the formatted table class. See tableType() for supported classes. Default is "gt".
- style Supported predefined styles. Check which ones with the function tableStyle().

Value

The code expression to style the table for as specific pre-defined style and table type.

Examples

```
tableStyleCode("gt")
tableStyleCode("flextable")
```

tableType	<i>Supported table classes</i>
-----------	--------------------------------

Description

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

Usage

```
tableType()
```

Value

A character vector of supported table types.

Examples

```
tableType()
```

themeVisOmap	<i>Apply a pre-defined visOmapResults theme to a ggplot</i>
--------------	---

Description

Apply a pre-defined `visOmapResults` theme to a `ggplot`

Usage

```
themeVisOmap(style = "default", fontsizeRef = 11)
```

Arguments

<code>style</code>	Character indicating which pre-defined style to apply. Refer to the function <code>plotStyle()</code> to see the available styles.
<code>fontsizeRef</code>	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	<i>Generate a formatted table from a <summarised_result></i>
--------------	--

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 = "gt",
  hide = character(),
  columnOrder = character(),
  factor = list(),
  style = "default",
  showMinCellCount = TRUE,
  .options = list()
)
```

Arguments

- | | |
|--------------|--|
| result | A <code><summarised_result></code> object. |
| estimateName | A named list of estimate names to join, sorted by computation order. Use <code><...></code> 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: <ul style="list-style-type: none"> Any of the columns returned by <code>tableColumns(result)</code> 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 <code>settingsColumns(result)</code> .
groupColumn	Columns to use as group labels, to see options use <code>tableColumns(result)</code> . 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: <code>list("newGroupName" = c("variable_name", "variable_level"))</code> . *tidy: The tidy format applied to column names replaces "_" with a space and converts to sentence case. Use <code>rename</code> to customise specific column names.
rename	A named vector to customise column names, e.g., <code>c("Database name" = "cdm_name")</code> . The function renames all column names not specified here into a tidy* format.
type	The desired format of the output table. See <code>tableType()</code> for allowed options.
hide	Columns to drop from the output table. By default, <code>result_id</code> and <code>estimate_type</code> are always dropped.
columnOrder	Character vector establishing the position of the columns in the formatted table. Columns in either header, <code>groupColumn</code> , or <code>hide</code> will be ignored.
factor	A named list where names refer to columns (see available columns in <code>tableColumns()</code>) 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	A character string or custom R code to define the visual formatting of the table. This argument can be provided in two ways: - Pre-defined Styles (Character String): Use a name for a built-in style (e.g., "darwin"). See <code>tableStyle()</code> for available options. - Custom Code (Advanced): Supply a block of custom R code. This code must be specific to the table type. See <code>tableStyleCode()</code> for structural examples.
showMinCellCount	If TRUE, suppressed estimates will be indicated with "<{min_cell_count}", otherwise, the default na defined in <code>.options</code> will be used.
.options	A named list with additional formatting options. <code>visOmopResults::tableOptions()</code> 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 |>
  visOmapTable(
    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 = "gt",
  hide = character(),
  style = "default",
  .options = list()
)

```

Arguments

result	A table to format.
estimateName	A named list of estimate names to join, sorted by computation order. Use <code><...></code> 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 <code>estimate_value</code> column to pivot the headers.

groupColumn	Columns to use as group labels, to see options use <code>tableColumns(result)</code> . 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: <code>list("newGroupName" = c("variable_name", "variable_level"))</code> . *tidy: The tidy format applied to column names replaces "_" with a space and converts to sentence case. Use <code>rename</code> to customise specific column names.
rename	A named vector to customise column names, e.g., <code>c("Database name" = "cdm_name")</code> . The function renames all column names not specified here into a tidy* format.
type	The desired format of the output table. See <code>tableType()</code> for allowed options.
hide	Columns to drop from the output table.
style	A character string or custom R code to define the visual formatting of the table. This argument can be provided in two ways: 1. Pre-defined Styles (Character String): Use a name for a built-in style (e.g., "darwin"). See <code>tableStyle()</code> for available options. 2. Custom Code (Advanced): Supply a block of custom R code. This code must be specific to the table type. See <code>tableStyleCode()</code> for structural examples.
.options	A named list with additional formatting options. <code>visOmapResults::tableOptions()</code> 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

- * **datasets**
 - data, [6](#)
- barPlot, [3](#)
- boxPlot, [4](#)
- customiseText, [5](#)
- data, [6](#)
- emptyPlot, [7](#)
- emptyTable, [7](#)
- formatEstimateName, [8](#)
- formatEstimateValue, [9](#)
- formatHeader, [10](#)
- formatMinCellCount, [11](#)
- formatTable, [11](#)
- mockSummarisedResult, [13](#)
- plotColumns, [14](#)
- plotStyle, [14](#)
- plotType, [15](#)
- requireExtrafont, [15](#)
- scatterPlot, [16](#)
- setGlobalPlotOptions, [17](#)
- setGlobalTableOptions, [18](#)
- tableColumns, [19](#)
- tableOptions, [19](#)
- tableStyle, [20](#)
- tableStyleCode, [20](#)
- tableType, [21](#)
- themeVisOmap, [21](#)
- visOmapTable, [22](#)
- visTable, [24](#)