Package 'visOmopResults'

January 15, 2025

Title Graphs and Tables for OMOP Results

Version 1.0.0

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 cli, dplyr, generics, glue, omopgenerics (>= 0.3.1), purrr, rlang, stringr, tidyr

Suggests covr, DT, flextable (>= 0.9.5), ggplot2, gt, htmltools, knitr, lifecycle, 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 [ctb] (<https://orcid.org/0000-0002-0847-4855>), Elin Rowlands [ctb] (<https://orcid.org/0009-0007-6629-4661>), Edward Burn [ctb] (<https://orcid.org/0000-0002-9286-1128>)

Repository CRAN

Date/Publication 2025-01-15 19:40:01 UTC

Contents

`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,
  facet = NULL,
  colour = 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.
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.
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* <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,
  label = character()
)
```

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.
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"))
```

---

emptyPlot *Returns an empty plot*

Description

Returns an empty plot

Usage

```
emptyPlot()
```

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 `tableType()` for allowed options.

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
  )
```

---

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. Not used for "datatable".
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".
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). Not used for "datatable".
groupOrder	Order in which to display group labels. Not used for "datatable".
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".

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"
  )
```

`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)
```

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

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,
  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 colors.
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)
```

`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` *Supported predefined styles for formatted tables*

Description

Supported predefined styles for formatted tables

Usage

```
tableStyle(type = "gt")
```

Arguments

`type`	Character string specifying the formatted table class. See `tableType()` for supported classes. Default is "gt".

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()
```

`themeVisOmop` *Apply visOmopResults default styling to a ggplot*

Description

Apply visOmopResults default styling to a ggplot

Usage

```
themeVisOmop(fontsizeRef = 10)
```

Arguments

`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` *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(),
  settingsColumn = character(),
  groupColumn = character(),
  rename = character(),
  type = "gt",
  hide = character(),
  columnOrder = character(),
  factor = list(),
  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	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.
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.

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 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

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 customise specific column names.
rename	A named vector to customise 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