Package ‘visOmopResults’

April 2, 2024

Title Graphs and Tables for OMOP Results

Version 0.2.1

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://oxford-pharmacoepi.github.io/visOmopResults/

BugReports https://github.com/oxford-pharmacoepi/visOmopResults/issues

Imports cli, dplyr, generics, glue, lifecycle, omopgenerics (>= 0.1.0), rlang, stringr, tidyr

Suggests flextable (>= 0.9.5), gt, officer, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

RoxygenNote 7.3.1

NeedsCompilation no

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

Repository CRAN

Date/Publication 2024-04-02 21:00:02 UTC

R topics documented:

`additionalColumns` *Identify additional columns in an omop result object*

Description

Identifies and returns the unique values in additional_name column.

Usage

```
additionalColumns(result, overall = lifecycle::deprecated())
```

Arguments

`result`	A summarised_result.
`overall`	deprecated.

Value

Unique values of the additional name column.

Examples

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

`appendSettings` *Append settings as rows in a summarised result*

Description

[Experimental]

Usage

```
appendSettings(x, colsSettings)
```

Arguments

`x`	A tibble with columns from summarised result and columns corresponding to settings.
`colsSettings`	Columns of x to append as settings rows.

Value

A tibble.

Examples

```
result <- mockSummarisedResult()[1,] |>
dplyr::mutate(
  mock_default = TRUE,
  example_setting = 1
)
appendSettings(result, colsSettings = c("mock_default", "example_setting"))
```

`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,
  estimateNameFormat = NULL,
  keepNotFormatted = TRUE,
  useFormatOrder = TRUE
)
```

Arguments

`result`	A summarised_result.
`estimateNameFormat`	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 estimateNameFormat (TRUE), or use the order in the input dataframe (FALSE).

Value

A summarised_result object.

Examples

```
result <- mockSummarisedResult()
result |>
  formatEstimateName(
    estimateNameFormat = 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 gtTable and fxTable functions.

Usage

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

Arguments

`result`	A summarised_result.
`header`	Names of the columns to make headers. Names that doesn't correspond to a column of the table result, will be used as headers at the defined position.
`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` *Format a summarised_result object into a gt, flextable or tibble object*

Description

Format a summarised_result object into a gt, flextable or tibble object

Usage

```
formatTable(
  result,
  formatEstimateName,
  header,
  split,
  groupColumn = NULL,
  type = "gt",
  minCellCount = 5,
 excludeColumns = c("result_id", "result_type", "package_name", "package_version",
    "estimate_type"),
  .options = list()
)
```

Arguments

`result`	A summarised_result.
`formatEstimateName`	Named list of estimate name's to join, sorted by computation order. Indicate estimate_name's between <...>.
`header`	A vector containing which elements should go into the header in order (`cdm_name`, `group`, `strata`, `additional`, `variable`, `estimate`, and `settings`).
`split`	A vector containing the name-level groups to split ("group", "strata", "additional"), or an empty character vector to not split.
`groupColumn`	Column to use as group labels.
`type`	Type of desired formatted table, possibilities: "gt", "flextable", "tibble".
`minCellCount`	Counts below which results will be clouded.
`excludeColumns`	Columns to drop from the output table.
`.options`	Named list with additional formatting options. visOmopResults::optionsFormatTable() shows allowed arguments and their default values.

Value

A tibble, gt, or flextable object.

Examples

```
mockSummarisedResult() |> formatTable(
  formatEstimateName = c("N%" = "<count> (<percentage>)",
                         "N" = "<count>",
                         "Mean (SD)" = "<mean> (<sd>)"),
  header = c("group"),
  split = c("group","strata",  "additional")
)
```

`fxTable` *Creates a flextable 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

```
fxTable(
  x,
  delim = "\n",
  style = "default",
  na = "-",
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  groupNameCol = NULL,
  groupNameAsColumn = 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 table. Accepted 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 flextable style.
`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).
`groupNameCol`	Column to use as group labels.
`groupNameAsColumn`	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.

Examples

```
mockSummarisedResult() |>
  formatEstimateValue(decimals = c(integer = 0, numeric = 1)) |>
  formatHeader(header = c("Study strata", "strata_name", "strata_level"),
               includeHeaderName = FALSE) |>
  fxTable(
    style = "default",
    na = "--",
    title = "fxTable example",
    subtitle = NULL,
    caption = NULL,
    groupNameCol = "group_level",
    groupNameAsColumn = TRUE,
    groupOrder = c("cohort1", "cohort2"),
    colsToMergeRows = "all_columns"
 )
```

`groupColumns` *Identify group columns in an omop result object*

Description

Identifies and returns the unique values in group_name column.

Usage

```
groupColumns(result, overall = lifecycle::deprecated())
```

Arguments

`result`	A summarised_result.
`overall`	deprecated.

Value

Unique values of the group name column.

Examples

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

`gtTable` *Creates a 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

```
gtTable(
  x,
  delim = "\n",
  style = "default",
  na = "-",
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  groupNameCol = NULL,
  groupNameAsColumn = 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 table. Accepted 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 style

`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).
`groupNameCol`	Column to use as group labels.
`groupNameAsColumn`	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.

Examples

```
mockSummarisedResult() |>
  formatEstimateValue(decimals = c(integer = 0, numeric = 1)) |>
  formatHeader(header = c("Study strata", "strata_name", "strata_level"),
              includeHeaderName = FALSE) |>
  gtTable(
    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,
    groupNameCol = "group_level",
    groupNameAsColumn = FALSE,
    groupOrder = c("cohort1", "cohort2"),
    colsToMergeRows = "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(settings = FALSE)
```

Arguments

`settings`	If TRUE settings will be appended.

Value

An object of the class summarised_result with mock data.

Examples

```
mockSummarisedResult()
```

`optionsFormatTable` *Additional arguments for the function formatTable*

Description

It provides a list of allowed inputs for .option argument in formatTable and their given default value.

Usage

```
optionsFormatTable()
```

Value

The default .options named list.

Examples

```
{
optionsFormatTable()
}
```

`pivotEstimates` *Set estimates as columns*

Description

[Experimental] 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()
```

`pivotSettings` *Add settings as columns*

Description

[Experimental] Pivot settings rows into columns.

Usage

```
pivotSettings(result)
```

Arguments

`result`	A summarised_result.

Value

A tibble.

Examples

```
result <- mockSummarisedResult(settings = TRUE)
result |> pivotSettings()
```

---

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",
  overall = lifecycle::deprecated()
)
```

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.
`overall`	deprecated.

Value

A dataframe.

Examples

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

splitAll *Split group, strata and additional at once.*

Description

Pivots the input dataframe so group, strata and additional name columns are transformed into columns that contain values from the corresponding level columns (group, strata, and additional).

Usage

```
splitAll(
  result,
  keep = FALSE,
  fill = "overall",
  overall = lifecycle::deprecated()
)
```

Arguments

result	A summarised_result object.
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.
overall	deprecated.

Value

A dataframe with group, strata and additional name 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",
  overall = lifecycle::deprecated()
)
```

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.
`overall`	deprecated.

Value

A dataframe.

Examples

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

`splitNameLevel` *Split name and level columns into the columns*

Description

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",
  overall = lifecycle::deprecated()
)
```

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.
`overall`	deprecated.

Value

A dataframe with the specified name column values as columns.

Examples

```
mockSummarisedResult() |>
  splitNameLevel(name = "group_name",
                 level = "group_level",
                 keep = FALSE)
```

`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",
  overall = lifecycle::deprecated()
)
```

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.
`overall`	deprecated.

Value

A dataframe.

Examples

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

`strataColumns` *Identify strata columns in an omop result object*

Description

Identifies and returns the unique values in strata_name column.

Usage

```
strataColumns(result, overall = lifecycle::deprecated())
```

Arguments

`result`	A summarised_result.
`overall`	deprecated.

Value

Unique values of the strata name column.

Examples

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

`tidy.summarised_result` *Get a tidy visualization of a summarised_result object*

Description

[Experimental] Provides tools for obtaining a tidy version of a summarised_result object. If the summarised results object contains settings, these will be transformed into columns.

Usage

```
## S3 method for class 'summarised_result'
tidy(
  x,
  splitGroup = TRUE,
  splitStrata = TRUE,
  splitAdditional = TRUE,
  pivotEstimatesBy = "estimate_name",
  nameStyle = NULL,
  ...
)
```

Arguments

`x`	A summarised_result.
`splitGroup`	If TRUE it will split the group name-level column pair.
`splitStrata`	If TRUE it will split the group name-level column pair.
`splitAdditional`	If TRUE it will split the group name-level column pair.
`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.
`...`	For compatibility (not used).

Value

A tibble.

Examples

```
result <- mockSummarisedResult()

result |> tidy()
```

`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

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.

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 |>
  uniteNameLevel(
    cols = c("sex", "age_group"),
    name = "new_column_name",
    level = "new_column_level"
  )
```

`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"))
```

Index