Package ‘visOmopResults’

May 2, 2024

Title Graphs and Tables for OMOP Results

Version 0.3.0

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

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

License Apache License (>= 2)

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

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

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

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

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

RoxygenNote 7.3.1

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>)

Repository CRAN

Date/Publication 2024-05-02 14:40:03 UTC
additionalColumns

Identify additional columns in an omop result object

Description
Identifies and returns the unique values in additional_name column.

Usage
additionalColumns(result)

Arguments
result A summarised_result.

Value
Unique values of the additional name column.
**addSettings**

Add settings columns to a summaries_result object.

**Description**

Add settings columns to a summaries_result object.

**Usage**

```r
addSettings(result, columns = NULL)
```

**Arguments**

- `result`: A summarised_result object.
- `columns`: Settings to be added as columns, by default all settings will be added.

**Value**

A summarised_result object with the added setting columns.

---

**filterSettings**

Filter a summarised_result

**Description**

Filter a summarised_result

**Usage**

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

**Arguments**

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

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

Examples

```r
library(tibble)
library(omopgenerics)

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

x

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

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

```r
formatEstimateName(
  result, 
  estimateNameFormat = NULL, 
  keepNotFormatted = TRUE, 
  useFormatOrder = TRUE
)
```
formatEstimateName

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 estimateName-Format (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 function

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

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

**fxTable**

Createss a fletable object from a dataframe

**Description**

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

**Usage**

```r
fxTable(
  x,
  delim = "\n",
  style = "default",
  na = "-",
  title = NULL,
  subtitle = NULL,
  caption = NULL,
  groupColumn = NULL,
  groupNameCol = lifecycle::deprecated(),
  groupAsColumn = FALSE,
  groupNameAsColumn = lifecycle::deprecated(),
  groupOrder = NULL,
  colsToMergeRows = NULL
)
```
## Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>x</strong></td>
<td>A dataframe.</td>
</tr>
<tr>
<td><strong>delim</strong></td>
<td>Delimiter.</td>
</tr>
<tr>
<td><strong>style</strong></td>
<td>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 &quot;default&quot; to get visOmopResults style, or NULL for flextable style.</td>
</tr>
<tr>
<td><strong>na</strong></td>
<td>How to display missing values.</td>
</tr>
<tr>
<td><strong>title</strong></td>
<td>Title of the table, or NULL for no title.</td>
</tr>
<tr>
<td><strong>subtitle</strong></td>
<td>Subtitle of the table, or NULL for no subtitle.</td>
</tr>
<tr>
<td><strong>caption</strong></td>
<td>Caption for the table, or NULL for no caption. Text in markdown formatting style (e.g. <em>Your caption here</em> for caption in italics).</td>
</tr>
<tr>
<td><strong>groupColumn</strong></td>
<td>Column to use as group labels.</td>
</tr>
<tr>
<td><strong>groupNameCol</strong></td>
<td>[Deprecated] This argument was renamed to &quot;groupColumn&quot; for consistency throughout the package functions.</td>
</tr>
<tr>
<td><strong>groupAsColumn</strong></td>
<td>Whether to display the group labels as a column (TRUE) or rows (FALSE).</td>
</tr>
<tr>
<td><strong>groupNameAsColumn</strong></td>
<td>[Deprecated] This argument was renamed to &quot;groupAsColumn&quot; for consistency with the argument &quot;groupColumn&quot;.</td>
</tr>
<tr>
<td><strong>groupOrder</strong></td>
<td>Order in which to display group labels.</td>
</tr>
<tr>
<td><strong>colsToMergeRows</strong></td>
<td>Names of the columns to merge vertically when consecutive row cells have identical values. Alternatively, use &quot;all_columns&quot; to apply this merging to all columns, or use NULL to indicate no merging.</td>
</tr>
</tbody>
</table>

## Value

A flextable object.

A flextable object.

## Examples

```r
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,  
    groupColumn = "group_level",  
    groupAsColumn = 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**

```r
groupColumns(result)
```

**Arguments**

- `result` A summarised_result.

**Value**

Unique values of the group name column.

**Examples**

```r
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**

```r
gtTable(
  x,
  delim = "\n",
  style = "default",
  na = "-",
  title = NULL,
  subtitle = NULL,
  caption = 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).
- **groupColumn**: Column to use as group labels.
- **groupNameCol**: [Deprecated] This argument was renamed to "groupColumn" for consistency throughout the package functions.
- **groupByColumn**: Whether to display the group labels as a column (TRUE) or rows (FALSE).
- **groupNameAsColumn**: [Deprecated] This argument was renamed to "groupByColumn" for consistency with the argument "groupColumn".
- **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

```r
mockSummarisedResult() |> formatEstimateValue(decimals = c(integer = 0, numeric = 1)) |> formatHeader(header = c("Study strata", "strata_name", "strata_level"), includeHeaderName = FALSE) |> gtTable(
```
mockSummarisedResult

A summarised_result object filled with mock data

Description

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

Usage

mockSummarisedResult()

Value

An object of the class summarised_result with mock data.

Examples

mockSummarisedResult()
optionsVisOmopTable  Additional arguments for the function visOmopTable

Description

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

Usage

optionsVisOmopTable()

Value

The default .options named list.

Examples

{
  optionsVisOmopTable()
}

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 pivoted.
nameStyle  Name style (glue package specifications) to customise names when pivoting estimates. If NULL standard tidyr::pivot_wider formatting will be used.

Value

A tibble.
Examples

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

---

**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

```r
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

```r
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.
**splitNameLevel**

**Usage**

```r
splitNameLevel(
  result,
  name = "group_name",
  level = "group_level",
  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**

```r
mockSummarisedResult() |>
  splitNameLevel()
```

---

**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**

```r
splitNameLevel(
  result,
  name = "group_name",
  level = "group_level",
  keep = FALSE,
  fill = "overall",
  overall = lifecycle::deprecated()
)
```
splitStrata

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.
strataColumns

Value

A dataframe.

Examples

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

---

**Description**

Identifies and returns the unique values in strata_name column.

**Usage**

```r
strataColumns(result)
```

**Arguments**

- `result` A summarised_result.

**Value**

Unique values of the strata name column.

**Examples**

```r
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

```r
## S3 method for class 'summarised_result'
tidy(
  x,
  splitGroup = TRUE,
  splitStrata = TRUE,
  splitAdditional = TRUE,
  addSettings = 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.
- `addSettings` Whether to add settings as columns or not.
- `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

```r
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

```r
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

```r
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

```r
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

```r
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

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

visOmopTable

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

Description

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

Usage

visOmopTable(
  result,
  formatEstimateName,  
  header,  
  split,  
  groupColumn = NULL,  
  type = "gt",  
  renameColumns = NULL,  
  showMinCellCount = TRUE,  
  minCellCount = lifecycle::deprecated(),  
  excludeColumns = c("result_id", "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**
Columns to use as group labels. By default the name of the new group will be the column names separated by ".". To specify a new grouping name enter a named list such as: list(newGroupName = c("variable_name", "variable_level"))

**type**
Type of desired formatted table, possibilities: "gt", "flextable", "tibble".

**renameColumns**
Named vector to customise column names, for instance: c("Database name" = "cdm_name"). By default column names are transformed to sentence case.

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

**minCellCount**
[Deprecated] Suppression of estimates when counts < minCellCount should be done before with omopgenerics::suppress().

**excludeColumns**
Columns to drop from the output table.

**.options**
Named list with additional formatting options. visOmopResults::optionsVisOmopTable() shows allowed arguments and their default values.

Value

A tibble, gt, or flextable object.

Examples

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

additionalColumns, 2
addSettings, 3

filterSettings, 3
formatEstimateName, 4
formatEstimateValue, 5
formatHeader, 6
fxTable, 7

groupColumns, 9
gtTable, 9

mockSummarisedResult, 11
optionsVisOmopTable, 12

pivotEstimates, 12

splitAdditional, 13
splitAll, 14
splitGroup, 14
splitNameLevel, 15
splitStrata, 16
strataColumns, 17

tidy.summarised_result, 17

uniteAdditional, 19
uniteGroup, 20
uniteNameLevel, 20
uniteStrata, 21

visOmopTable, 22