Package ‘tidyHeatmap’

May 7, 2020

Type Package

Title A Tidy Implementation of Heatmap

Version 0.99.18

Maintainer Stefano Mangiola <mangiolastefano@gmail.com>

Description This is a tidy implementation for heatmap. At the moment it is based on the (great) package 'ComplexHeatmap'. The goal of this package is to interface a tidy data frame with this powerful tool. Some of the advantages are: Row and/or columns colour annotations are easy to integrate just specifying one parameter (column names). Custom grouping of rows is easy to specify providing a grouped tbl. For example: df %>% group_by(...). Labels size adjusted by row and column total number. Default use of Brewer and Viridis palettes.

License GPL-3

URL https://www.r-project.org,
https://github.com/stemangiola/tidyHeatmap

BugReports https://github.com/stemangiola/tidyHeatmap

Depends R (>= 3.6),
ComplexHeatmap (>= 2.2.0)

Imports dplyr,
magrittr,
tidyr,
rlang,
purrr,
stats,
viridis,
circlize,
RColorBrewer,
grid,
grDevices,
utils,
lifecycle

Suggests testthat,
BiocManager,
knitr,
qpdf,
covr

VignetteBuilder knitr

RdMacros lifecycle

Biarch true

biocViews AssayDomain, Infrastructure

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

R topics documented:

`add_attr` *Add attribute to abject*

Description

Add attribute to abject

Usage

```
add_attr(var, attribute, name)
```

Arguments

`var`	A tibble
`attribute`	An object
`name`	A character name of the attribute

Value

A tibble with an additional attribute

`add_class` *Add class to abject*

Description

Add class to abject

Usage

```
add_class(var, name)
```

Arguments

`var`	A tibble
`name`	A character name of the attribute

Value

A tibble with an additional attribute

`as_matrix` *Get matrix from tibble*

Description

Get matrix from tibble

Usage

```
as_matrix(tbl, rownames = NULL, do_check = TRUE)
```

Arguments

`tbl`	A tibble
`rownames`	A character string of the rownames
`do_check`	A boolean

Value

A matrix

`check_if_counts_is_na` *Check whether there are NA counts*

Description

Check whether there are NA counts

Usage

```
check_if_counts_is_na(.data, .abundance)
```

Arguments

`.data`	A tibble of read counts
`.abundance`	A character name of the read count column

Value

A tbl

`check_if_duplicated_genes` *Check whether there are duplicated genes/transcripts*

Description

Check whether there are duplicated genes/transcripts

Usage

```
check_if_duplicated_genes(.data, .sample, .transcript, .abundance)
```

Arguments

`.data`	A tibble of read counts
`.sample`	A character name of the sample column
`.transcript`	A character name of the transcript/gene column
`.abundance`	A character name of the read count column

Value

A tbl

`check_if_wrong_input` *Check whether there are NA counts*

Description

Check whether there are NA counts

Usage

```
check_if_wrong_input(.data, list_input, expected_type)
```

Arguments

`.data`	A tibble of read counts
`list_input`	A list
`expected_type`	A character string

Value

A tbl

`drop_class` *Remove class to abject*

Description

Remove class to abject

Usage

```
drop_class(var, name)
```

Arguments

`var`	A tibble
`name`	A character name of the class

Value

A tibble with an additional attribute

error_if_log_transformed

Check whether a numeric vector has been log transformed

Description

Check whether a numeric vector has been log transformed

Usage

```
error_if_log_transformed(x, .abundance)
```

Arguments

x	A numeric vector
.abundance	A character name of the transcript/gene abundance column

Value

NA

get_abundance_norm_if_exists

Get column names either from user or from attributes

Description

Get column names either from user or from attributes

Usage

```
get_abundance_norm_if_exists(.data, .abundance)
```

Arguments

.data	A tibble
.abundance	A character name of the abundance column

Value

A list of column enquo or error

get_elements *Get column names either from user or from attributes*

Description

Get column names either from user or from attributes

Usage

```
get_elements(.data, .element, of_samples = TRUE)
```

Arguments

`.data`	A tibble
`.element`	A character name of the sample column
`of_samples`	A boolean

Value

A list of column enquo or error

get_elements_features *Get column names either from user or from attributes*

Description

Get column names either from user or from attributes

Usage

```
get_elements_features(.data, .element, .feature, of_samples = TRUE)
```

Arguments

`.data`	A tibble
`.element`	A character name of the sample column
`.feature`	A character name of the transcript/gene column
`of_samples`	A boolean

Value

A list of column enquo or error

get_elements_features_abundance *Get column names either from user or from attributes*

Description

Get column names either from user or from attributes

Usage

```
get_elements_features_abundance(
  .data,
  .element,
  .feature,
  .abundance,
  of_samples = TRUE
)
```

Arguments

.data	A tibble
.element	A character name of the sample column
.feature	A character name of the transcript/gene column
.abundance	A character name of the read count column
of_samples	A boolean

Value

A list of column enquo or error

get_sample_counts *Get column names either from user or from attributes*

Description

Get column names either from user or from attributes

Usage

```
get_sample_counts(.data, .sample, .abundance)
```

Arguments

.data	A tibble
.sample	A character name of the sample column
.abundance	A character name of the read count column

Value

A list of column enquo or error

`get_sample_transcript` *Get column names either from user or from attributes*

Description

Get column names either from user or from attributes

Usage

```
get_sample_transcript(.data, .sample, .transcript)
```

Arguments

`.data`	A tibble
`.sample`	A character name of the sample column
`.transcript`	A character name of the transcript/gene column

Value

A list of column enquo or error

`get_sample_transcript_counts` *Get column names either from user or from attributes*

Description

Get column names either from user or from attributes

Usage

```
get_sample_transcript_counts(.data, .sample, .transcript, .abundance)
```

Arguments

`.data`	A tibble
`.sample`	A character name of the sample column
`.transcript`	A character name of the transcript/gene column
`.abundance`	A character name of the read count column

Value

A list of column enquo or error

get_x_y_annotation_columns *get_x_y_annotation_columns*

Description

get_x_y_annotation_columns

Usage

```
get_x_y_annotation_columns(.data, .horizontal, .vertical, .abundance)
```

Arguments

`.data`	A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |
`.horizontal`	The name of the column horizontally presented in the heatmap
`.vertical`	The name of the column vertically presented in the heatmap
`.abundance`	The name of the transcript/gene abundance column

Value

A list

heatmap *Creates a ‘ComplexHeatmap‘ plot from ‘tbl_df‘*

Description

heatmap() takes a tbl object and easily produces a ComplexHeatmap plot, with integration with tibble and dplyr frameworks.

Usage

```
heatmap(
  .data,
  .row,
  .column,
  .value,
  annotation = NULL,
  transform = NULL,
  palette_value = c("#440154FF", "#21908CFF", "#fefada"),
  palette_discrete = list(),
  palette_continuous = list(),
  .abundance = NULL,
  .horizontal = NULL,
  .vertical = NULL,
  log_transform = NULL,
  palette_abundance = NULL,
  ...
)
```

Arguments

`.data` A ‘tbl_df‘ formatted as | <ELEMENT> | <FEATURE> | <VALUE> | <...> |

`.row` The name of the column vertically presented in the heatmap

`.column` The name of the column horizontally presented in the heatmap

`.value` The name of the transcript/gene abundance column

`annotation` Vector of quotes

`transform` A function, used to tranform .value, for example log

`palette_value` A character vector This is the palette that will be used as gradient for .value

`palette_discrete` A list of character vectors. This is the list of palettes that will be used for horizontal and vertical discrete annotations. The discrete classification of annotations depends on the column type of your input tibble (e.g., character and factor).

`palette_continuous` A list of character vectors. This is the list of palettes that will be used for horizontal and vertical continuous annotations. The continuous classification of annotations depends on the column type of your input tibble (e.g., integer, numerical, double).

`.abundance` DEPRECATED. Please use .value instead

`.horizontal` DEPRECATED. Please use .column instead

`.vertical` DEPRECATED. Please use .row instead

`log_transform` DEPRECATED. Please use transform instead

`palette_abundance` DEPRECATED. Please use palette_value instead

`...` Further arguments to be passed to ComplexHeatmap::Heatmap

Details

Maturing

To be added.

Value

A ‘ComplexHeatmap‘ object

Examples

```
library(dplyr)
tidyHeatmap::N52 %>%
group_by( `Cell type`) %>%
tidyHeatmap::heatmap(
 .row = symbol_ct,
 .column = UBR,
 .value = `read count normalised log`,
 annotation = CAPRA_TOTAL
)
```

ifelse2_pipe *This is a generalisation of ifelse that acceots an object and return an objects*

Description

This is a generalisation of ifelse that acceots an object and return an objects

Usage

```
ifelse2_pipe(.x, .p1, .p2, .f1, .f2, .f3 = NULL)
```

Arguments

`.x`	A tibble
`.p1`	A boolean
`.p2`	ELSE IF condition
`.f1`	A function
`.f2`	A function
`.f3`	A function

Value

A tibble

ifelse_pipe *This is a generalisation of ifelse that acceots an object and return an objects*

Description

This is a generalisation of ifelse that acceots an object and return an objects

Usage

```
ifelse_pipe(.x, .p, .f1, .f2 = NULL)
```

Arguments

`.x`	A tibble
`.p`	A boolean
`.f1`	A function
`.f2`	A function

Value

A tibble

`N52` *Example data set N52*

Description

Example data set N52

Usage

```
N52
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 520 rows and 15 columns.

`parse_formula` *.formula parser*

Description

.formula parser

Usage

```
parse_formula(fm)
```

Arguments

`fm`	a formula

Value

A character vector

`pasilla` *Example data set Pasilla*

Description

Example data set Pasilla

Usage

```
pasilla
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 504 rows and 7 columns.

plot_heatmap *plot_heatmap*

Description

plot_heatmap() takes a tbl object and easily produces a ComplexHeatmap plot, with integration with tibble and dplyr frameworks.

Usage

```
plot_heatmap(
  .data,
  .horizontal,
  .vertical,
  .abundance,
  annotation = NULL,
  transform = NULL,
  palette_abundance = c("#440154FF", "#21908CFF", "#fefada"),
  palette_discrete = list(),
  palette_continuous = list(),
  ...
)
```

Arguments

`.data` A ‘tbl‘ formatted as | <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |

`.horizontal` The name of the column horizontally presented in the heatmap

`.vertical` The name of the column vertically presented in the heatmap

`.abundance` The name of the transcript/gene abundance column

`annotation` Vector of quotes

`transform` A function, used to tranform .value, for example log

`palette_abundance` A character vector, or a function for higher customisation (colorRamp2). This is the palette that will be used as gradient for abundance. If palette_abundance is a vector of hexadecimal colous, it should have 3 values. If you want more customisation, you can pass to palette_abundance a function, that is derived as for example ‘colorRamp2(c(-2, 0, 2), palette_abundance)‘

`palette_discrete` A list of character vectors. This is the list of palettes that will be used for horizontal and vertical discrete annotations. The discrete classification of annotations depends on the column type of your input tibble (e.g., character and factor).

`palette_continuous` A list of character vectors. This is the list of palettes that will be used for horizontal and vertical continuous annotations. The continuous classification of annotations depends on the column type of your input tibble (e.g., integer, numerical, double).

`...` Further arguments to be passed to ComplexHeatmap::Heatmap

Details

To be added.

Value

A ‘ComplexHeatmap‘ object

`prepend` *From rlang deprecated*

Description

From rlang deprecated

Usage

```
prepend(x, values, before = 1)
```

Arguments

`x`	An array
`values`	An array
`before`	A boolean

Value

An array

`scale_design` *Scale design matrix*

Description

Scale design matrix

Usage

```
scale_design(df, .formula)
```

Arguments

`df`	A tibble
`.formula`	a formula

Value

A tibble

select_closest_pairs *Sub function of remove_redundancy_elements_though_reduced_dimensions*

Description

Sub function of remove_redundancy_elements_though_reduced_dimensions

Usage

```
select_closest_pairs(df)
```

Arguments

`df`	A tibble

Value

A tibble with pairs to drop

Index