Package ‘tidyHeatmap’

July 7, 2021

Type Package

Title A Tidy Implementation of Heatmap

Version 1.3.1

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)

Imports methods,

  stats,
  utils,
  dplyr (>= 0.8.5),
  magrittr (>= 1.5),
  tidyr (>= 1.0.3),
  rlang (>= 0.4.5),
  purrr (>= 0.3.3),
  tibble,
  ComplexHeatmap (>= 2.2.0),
  viridis (>= 0.5.1),
  circlize (>= 0.4.8),
  RColorBrewer (>= 1.1),
  grid,
  grDevices,
  lifecycle (>= 0.2.0),
  dendextend
Suggests spelling,
testthat,
vdiffr,
BiocManager,
knitr,
rmatch,
qpdf,
covr,
roxygen2

VignetteBuilder knitr
RdMacros lifecycle
Biarch true
biocViews AssayDomain, Infrastructure
Encoding UTF-8
LazyData true
RoxygenNote 7.1.1
Language en-US

R topics documented:

add_annotation .................................................. 3
add_attr .......................................................... 4
add_bar ............................................................ 4
add_bar,InputHeatmap-method ................................. 5
add_class .......................................................... 6
add_line ............................................................ 6
add_line,InputHeatmap-method ............................... 7
add_point .......................................................... 8
add_tile ........................................................... 9
annot_to_list ...................................................... 10
as_matrix .......................................................... 10
check_if_counts_is_na ......................................... 11
check_if_duplicated_genes .................................... 11
check_if_wrong_input ........................................... 12
drop_class ......................................................... 12
derror_if_log_transformed .................................... 13
get_abundance_norm_if_exists ................................ 13
get_elements ...................................................... 14
get_elements_features ......................................... 14
get_elements_features_abundance ............................ 15
get_sample_counts .............................................. 15
get_sample_transcript .......................................... 16
get_sample_transcript_counts ................................ 16
get_x_y_annotation_columns .................................. 17
heatmap ............................................................. 17
ifelse2_pipe ..................................................... 20
ifelse_pipe ....................................................... 20
input_heatmap ..................................................... 21
layer_arrow_up .................................................... 22
Description

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

Usage

```r
add_annotation(
  my_input_heatmap,
  annotation,
  type = rep("tile", length(quo_names(annotation))),
  palette_discrete = list(),
  palette_continuous = list()
)
```

Arguments

- `my_input_heatmap`
  A `InputHeatmap` formatted as | SAMPLE | TRANSCRIPT | COUNT | | ... |
- `annotation`
  Vector of quotes
- `type`
  A character vector of the set c("tile", "point", "bar", "line")
- `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).
Details
To be added.

Value
A ‘ComplexHeatmap’ object

---

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_bar
Adds a bar annotation layer to a ‘InputHeatmap’, that on evaluation creates a ‘ComplexHeatmap’

Description
add_bar() from a ‘InputHeatmap’ object, adds a bar annotation layer.

Usage
add_bar(.data, .column, palette = NULL)

Arguments
- .data: A ‘tbl_df’ formatted as | <ELEMENT> | <FEATURE> | <VALUE> | <...> |
- .column: Vector of quotes
- palette: A character vector of colors 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).
add_bar, InputHeatmap-method

Details

[Maturing]
It uses ‘ComplexHeatmap’ as visualisation tool.

Value

A ‘InputHeatmap’ object that gets evaluated to a ‘ComplexHeatmap’

Examples

library(dplyr)

hm =
tidyHeatmap::N52 %>%
tidyHeatmap::heatmap(
  .row = symbol_ct,
  .column = UBR,
  .value = `read count normalised log`
)

hm %>% add_bar()
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

---

add_line  

*Adds a line annotation layer to a 'InputHeatmap', that on evaluation creates a 'ComplexHeatmap'*

**Description**

`add_line()` from a 'InputHeatmap' object, adds a line annotation layer.

**Usage**

`add_line(.data, .column, palette = NULL)`

**Arguments**

- **.data**  
  A 'tbl_df' formatted as | <ELEMENT> | <FEATURE> | <VALUE> | <...> |

- **.column**  
  Vector of quotes

- **palette**  
  A character vector of colors 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).

**Details**

[Maturing]

It uses 'ComplexHeatmap' as visualisation tool.

**Value**

A 'InputHeatmap' object that gets evaluated to a 'ComplexHeatmap'
Examples

```r
library(dplyr)

hm =
tidyHeatmap::N52 %>%
tidyHeatmap::heatmap(
   .row = symbol_ct,
   .column = UBR,
   .value = "read count normalised log"
)

hm %>% add_line()
```

Description

add_line

Usage

```r
## S4 method for signature 'InputHeatmap'
add_line(.data, .column, palette = NULL)
```

Arguments

- `.data`
  - A `tbl_df` formatted as `| <ELEMENT> | <FEATURE> | <VALUE> | <...> |`
- `.column`
  - Vector of quotes
- `.palette`
  - A character vector of colors 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).

Value

A `InputHeatmap` object that gets evaluated to a `ComplexHeatmap`
**add_point**

*Adds a point annotation layer to a ‘InputHeatmap’, that on evaluation creates a ‘ComplexHeatmap’*

**Description**

`add_point()` from a ‘InputHeatmap’ object, adds a point annotation layer.

**Usage**

```r
add_point(.data, .column, palette = NULL)
```

```r
# S4 method for signature 'InputHeatmap'
add_point(.data, .column, palette = NULL)
```

**Arguments**

- `.data`: A `tbl_df` formatted as `| <ELEMENT> | <FEATURE> | <VALUE> | <...> |`
- `.column`: Vector of quotes
- `palette`: A character vector of colors. 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).

**Details**

[Maturing]

It uses ‘ComplexHeatmap’ as visualisation tool.

**Value**

A ‘InputHeatmap’ object that gets evaluated to a ‘ComplexHeatmap’

A ‘InputHeatmap’ object that gets evaluated to a ‘ComplexHeatmap’

**Examples**

```r
library(dplyr)

hm = tidyHeatmap::N52 %>%
tidyHeatmap::heatmap(
  .row = symbol_ct,
  .column = UBR,
  .value = `read count normalised log`
)

hm %>% add_point()
```
Description

add_tile() from a ‘InputHeatmap’ object, adds a tile annotation layer.

Usage

add_tile(.data, .column, palette = NULL)

## S4 method for signature 'InputHeatmap'
add_tile(.data, .column, palette = NULL)

Arguments

.data A ‘tbl_df’ formatted as | <ELEMENT> | <FEATURE> | <VALUE> | <...> |
.column Vector of quotes
.palette A character vector of colors 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).

Details

[Maturing]
It uses ‘ComplexHeatmap’ as visualisation tool.

Value

A ‘InputHeatmap’ object that gets evaluated to a ‘ComplexHeatmap’
A ‘InputHeatmap’ object that gets evaluated to a ‘ComplexHeatmap’

Examples

library(dplyr)

hm =
tidyHeatmap::N52 %>%
tidyHeatmap::heatmap(
  .row = symbol_ct,
  .column = UBR,
  .value = `read count normalised log`
)

hm %>% add_tile(CAPRA>Total)
annot_to_list

Description
annot_to_list

Usage
annot_to_list(.data)

Arguments
.data A data frame

Value
A list

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**

```r
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**

```r
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**

```r
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**

```r
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**

**Description**

get_x_y_annotation_columns

**Usage**

get_x_y_annotation_columns(.data, .column, .row, .abundance)

**Arguments**

- **.data**: A 'tbl' formatted as `| <SAMPLE> | <TRANSCRIPT> | <COUNT> | <...> |
- **.column**: The name of the column horizontally presented in the heatmap
- **.row**: 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 'InputHeatmap' object from 'tbl_df' on evaluation creates a 'ComplexHeatmap'

**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,
  transform = NULL,
  .scale = "row",
  palette_value = c("#440154FF", "#21908CFF", "#fefada"),
  palette_grouping = list(),
  annotation = NULL,
  type = rep("tile", length(quo_names(annotation))),
  palette_discrete = list(),
  palette_continuous = list(),
  ...
)
heatmap_(
  .data,
  .row,
  .column,
  .value,
  transform = NULL,
  .scale = "row",
  palette_value = c("#440154FF", "#21908CFF", "#fefada"),
  palette_grouping = list(),
  annotation = NULL,
  type = rep("tile", length(quo_names(annotation))),
  palette_discrete = list(),
  palette_continuous = list(),
  ...
)

## S4 method for signature 'tbl'
heatmap(
  .data,
  .row,
  .column,
  .value,
  transform = NULL,
  .scale = "row",
  palette_value = c("#440154FF", "#21908CFF", "#fefada"),
  palette_grouping = list(),
  annotation = NULL,
  type = rep("tile", length(quo_names(annotation))),
  palette_discrete = list(),
  palette_continuous = list(),
  ...
)

## S4 method for signature 'tbl_df'
heatmap(
  .data,
  .row,
  .column,
  .value,
  transform = NULL,
  .scale = "row",
  palette_value = c("#440154FF", "#21908CFF", "#fefada"),
  palette_grouping = list(),
  annotation = NULL,
  type = rep("tile", length(quo_names(annotation))),
  palette_discrete = list(),
  palette_continuous = list(),
  ...
)
**heatmap**

**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 column for the value of the element/feature pair

`transform` A function, used to transform `.value` row-wise (e.g., `transform = log1p`)

`.scale` A character string. Possible values are c("none", "row", "column", "both")

`palette_value` A character vector. This is the palette that will be used as gradient for `.value`. For example c("red", "white", "blue"). For higher flexibility you can use `circlize::colorRamp2(c(-2, -1, 0, 1, 2), viridis::magma5)``

`palette_grouping` A list of character vectors. This is the list of palettes that will be used for grouping. For example `list(RColorBrewer::brewer.pal(8, "Accent"))` or `list(c("#B3E2CD", "#FDCDAC", "#CBD5E8"))` or `list(c("black", "red"))`

`annotation` DEPRECATED. please use the annotation functions `add_*` function (e.g., `tile`, `point`, `bar`, `line`).

`type` DEPRECATED. please use the annotation functions `add_*` function (e.g., `tile`, `point`, `bar`, `line`).

`palette_discrete` DEPRECATED. please use the annotation functions `add_*` function (e.g., `tile`, `point`, `bar`, `line`).

`palette_continuous` DEPRECATED. please use the annotation functions `add_*` function (e.g., `tile`, `point`, `bar`, `line`).

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

**Details**

[Maturing]

This function takes a `tbl` as an input and creates a `ComplexHeatmap` plot. The information is stored in a `InputHeatmap` object that is updated along the pipe statement, for example adding annotation layers.

**Value**

A `InputHeatmap` object that gets evaluated to a `ComplexHeatmap` object

A `InputHeatmap` object

A `InputHeatmap` object

A `InputHeatmap` object

**Examples**

```r
library(dplyr)

tidyHeatmap::N52 %>%
  group_by( 'Cell type') %>%
  tidyHeatmap::heatmap{
```
ifelse2_pipe

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

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

Description
This is a generalisation of ifelse that accepts 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

---

**Description**

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

**Usage**

```r
input_heatmap(
  .data,
  .horizontal,
  .vertical,
  .abundance,
  transform = NULL,
  .scale = "row",
  palette_value = c("#440154FF", "#21908CFF", "#fefada"),
  palette_grouping = list(),
  ...)
```

**Arguments**

- `.data` A `tib` 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
- `transform` A function, used to transform `.value`, for example log1p
- `.scale` A character string. Possible values are `c("none", "row", "column", "both")`
- `palette_value` A character vector, or a function for higher customisation (colorRamp2). This is the palette that will be used as gradient for abundance. If `palette_value` is a vector of hexadecimal colours, it should have 3 values. If you want more customisation, you can pass to `palette_value` a function, that is derived as for example `colorRamp2(c(-2, 0, 2), palette_value)`
- `palette_grouping` A list of character vectors. This is the list of palettes that will be used for grouping
- `...` Further arguments to be passed to ComplexHeatmap::Heatmap

**Details**

To be added.

**Value**

A `ComplexHeatmap` object
layer_arrow_up

Adds a layers of symbols above the heatmap tiles to a ‘InputHeatmap’, that on evaluation creates a ‘ComplexHeatmap’

Description

layer_arrow_up() from a ‘InputHeatmap’ object, adds a bar annotation layer.
layer_arrow_down() from a ‘InputHeatmap’ object, adds a bar annotation layer.
layer_point() from a ‘InputHeatmap’ object, adds a bar annotation layer.
layer_square() from a ‘InputHeatmap’ object, adds a bar annotation layer.
layer_diamond() from a ‘InputHeatmap’ object, adds a bar annotation layer.

Usage

layer_arrow_up(.data, ...)

## S4 method for signature 'InputHeatmap'
layer_arrow_up(.data, ...)

layer_arrow_down(.data, ...)

## S4 method for signature 'InputHeatmap'
layer_arrow_down(.data, ...)

layer_point(.data, ...)

## S4 method for signature 'InputHeatmap'
layer_point(.data, ...)

layer_square(.data, ...)

## S4 method for signature 'InputHeatmap'
layer_square(.data, ...)

layer_diamond(.data, ...)

## S4 method for signature 'InputHeatmap'
layer_diamond(.data, ...)

Arguments

.data A ‘InputHeatmap’

... Expressions that return a logical value, 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.
Details

[Maturing] It uses ‘ComplexHeatmap’ as visualisation tool.

[Maturing] It uses ‘ComplexHeatmap’ as visualisation tool.

[Maturing] It uses ‘ComplexHeatmap’ as visualisation tool.

[Maturing] It uses ‘ComplexHeatmap’ as visualisation tool.

[Maturing] It uses ‘ComplexHeatmap’ as visualisation tool.

[Maturing] It uses ‘ComplexHeatmap’ as visualisation tool.

Value

A ‘InputHeatmap’ object that gets evaluated to a ‘ComplexHeatmap’
A ‘InputHeatmap’ object that gets evaluated to a ‘ComplexHeatmap’
A ‘InputHeatmap’ object that gets evaluated to a ‘ComplexHeatmap’
A ‘InputHeatmap’ object that gets evaluated to a ‘ComplexHeatmap’
A ‘InputHeatmap’ object that gets evaluated to a ‘ComplexHeatmap’
A ‘InputHeatmap’ object that gets evaluated to a ‘ComplexHeatmap’

Examples

library(dplyr)

hm = tidyHeatmap:::N52 %>%
tidyHeatmap::heatmap(  
.row = symbol_ct,  
.column = UBR,  
.value = `read count normalised log`
)

hm %>% layer_arrow_up()

library(dplyr)

hm = tidyHeatmap:::N52 %>%
tidyHeatmap:::heatmap(  
.row = symbol_ct,
Example data set N52

Description

Example data set N52
**parse_formula**

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

```r
parse_formula(fm)
```

Arguments

- `fm` a formula

Value

A character vector

---

**pasilla Example data set Pasilla**

---

Description

Example data set Pasilla

Usage

```r
pasilla
```

Format

An object of class `tbl_df` (inherits from `tbl.data.frame`) with 504 rows and 8 columns.
**prepend**  
*From rlang deprecated*

**Description**

From rlang deprecated

**Usage**

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

**Arguments**

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

**Value**

An array

---

**quo_names**  
*Convert array of quosure (e.g. c(col_a, col_b)) into character vector*

**Description**

Convert array of quosure (e.g. c(col_a, col_b)) into character vector

**Usage**

```r
quo_names(v)
```

**Arguments**

- `v`  
  A array of quosures (e.g. c(col_a, col_b))

**Value**

A character vector
save_pdf

Save plot on PDF file

Description

save_pdf() takes as input a Heatmap from ComplexHeatmap and save it to PDF file

Usage

save_pdf(.heatmap, filename, width = NULL, height = NULL, units = c("in", "cm", "mm"))

Arguments

.heatmap  A 'Heatmap'
filename   A character string. The name of the output file/path
width     A 'double'. Plot width
height    A 'double'. Plot height
units     A character string. units ("in", "cm", or "mm")

Details

[Maturing]
It simply save an ‘Heatmap’ to a PDF file use pdf() function in the back end

Value

NA

Examples

library(dplyr)
tidyHeatmap::heatmap(
  dplyr::group_by(tidyHeatmap::pasilla, location, type),
  .column = sample,
  .row = symbol,
  .value = 'count normalised adjusted',
) %>%
save_pdf(tempfile())
save_pdf, Heatmap-method

Description

save_pdf

Usage

## S4 method for signature 'Heatmap'
save_pdf(  
  .heatmap,
  filename,
  width = NULL,
  height = NULL,
  units = c("in", "cm", "mm")
)

Arguments

- `.heatmap`: A 'Heatmap'
- `filename`: A character string. The name of the output file/path
- `width`: A 'double'. Plot width
- `height`: A 'double'. Plot height
- `units`: A character string. units ("in", "cm", or "mm")

save_pdf, InputHeatmap-method

Description

save_pdf

Usage

## S4 method for signature 'InputHeatmap'
save_pdf(  
  .heatmap,
  filename,
  width = NULL,
  height = NULL,
  units = c("in", "cm", "mm")
)
**scale_design**

**Arguments**

- `.heatmap` A 'Heatmap'
- `filename` A character string. The name of the output file/path
- `width` A 'double'. Plot width
- `height` A 'double'. Plot height
- `units` A character string. units ("in", "cm", or "mm")

---

**Description**

Scale design matrix

**Usage**

`scale_design(df, .formula)`

**Arguments**

- `df` A tibble
- `.formula` a formula

**Value**

A tibble

---

**scale_robust**

Scale counts in a robust way against sd == 0

**Description**

Scale counts in a robust way against sd == 0

**Usage**

`scale_robust(y)`

**Arguments**

- `y` A numerical array

**Value**

A scaled and centred numerical array
select_closest_pairs

*Sub function of remove_redundancy_elements_though_reduced_dimensions*

**Description**

Sub function of remove_redundancy_elements_though_reduced_dimensions

**Usage**

```r
select_closest_pairs(df)
```

**Arguments**

- **df**
  - A tibble

**Value**

A tibble with pairs to drop

split_rows

*Split the heatmap row-wise depending on the biggest branches in the cladogram.*

**Description**

split_rows() from a 'InputHeatmap' object, split the row cladogram.

split_columns() from a 'InputHeatmap' object, split the column cladogram.

**Usage**

```r
split_rows(.data, number_of_groups)
```

```r
## S4 method for signature 'InputHeatmap'
split_rows(.data, number_of_groups)
```

```r
split_columns(.data, number_of_groups)
```

```r
## S4 method for signature 'InputHeatmap'
split_columns(.data, number_of_groups)
```

**Arguments**

- **.data**
  - A 'InputHeatmap'

- **number_of_groups**
  - An integer. The number of groups to split the cladogram into.
split_rows

Details

[Maturing]
It uses 'ComplexHeatmap' as visualisation tool.

[Maturing]
It uses 'ComplexHeatmap' as visualisation tool.

Value

A 'InputHeatmap' object that gets evaluated to a 'ComplexHeatmap'

Examples

```r
library(dplyr)

hm <-
tidyHeatmap::N52 %>%
tidyHeatmap::heatmap(
  .row = symbol_ct,
  .column = UBR,
  .value = `read count normalised log`
)

hm %>% split_rows(2)
```

```r
library(dplyr)

hm <-
tidyHeatmap::N52 %>%
tidyHeatmap::heatmap(
  .row = symbol_ct,
  .column = UBR,
  .value = `read count normalised log`
)

hm %>% split_columns(2)
```
Index

* datasets
  N52, 24
  pasilla, 25
add_annotation, 3
add_attr, 4
add_bar, 4
add_bar, InputHeatmap-method, 5
add_class, 6
add_line, 6
add_line, InputHeatmap-method, 7
add_point, 8
add_point, InputHeatmap-method (add_point), 8
add_tile, 9
add_tile, InputHeatmap-method (add_tile), 9
annot_to_list, 10
as_matrix, 10
check_if_counts_is_na, 11
check_if_duplicated_genes, 11
check_if_wrong_input, 12
drop_class, 12
error_if_log_transformed, 13
get_abundance_norm_if_exists, 13
get_elements, 14
get_elements_features, 14
get_elements_features_abundance, 15
get_sample_counts, 15
get_sample_transcript, 16
get_sample_transcript_counts, 16
get_x_y_annotation_columns, 17
heatmap, 17
heatmap, tbl-method (heatmap), 17
heatmap, tbl_df-method (heatmap), 17
heatmap_(heatmap), 17
ifelse2_pipe, 20
input_heatmap, 21
layer_arrow_down (layer_arrow_up), 22
layer_arrow_down, InputHeatmap-method (layer_arrow_up), 22
layer_arrow_up, 22
layer_arrow_up, InputHeatmap-method (layer_arrow_up), 22
layer_diamond (layer_arrow_up), 22
layer_diamond, InputHeatmap-method (layer_arrow_up), 22
layer_point (layer_arrow_up), 22
layer_point, InputHeatmap-method (layer_arrow_up), 22
layer_square (layer_arrow_up), 22
layer_square, InputHeatmap-method (layer_arrow_up), 22
N52, 24
parse_formula, 25
pasilla, 25
prepend, 26
quo_names, 26
save_pdf, 27
save_pdf, Heatmap-method, 28
save_pdf, InputHeatmap-method, 28
scale_design, 29
scale_robust, 29
select_closest_pairs, 30
split_columns (split_rows), 30
split_columns, InputHeatmap-method (split_rows), 30
split_rows, 30
split_rows, InputHeatmap-method (split_rows), 30