Package ‘tidyHeatmap’

May 20, 2022

Type Package

Title A Tidy Implementation of Heatmap

Version 1.8.1

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),
tidyrr (>= 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,
patchwork
Suggests: spelling, testthat, vdiffr, BiocManager, knitr, rmarkdown, qpdf, covr, roxygen2, forcats, ggplot2

VignetteBuilder knitr
RdMacros lifecycle
Biarch true
biocViews AssayDomain, Infrastructure
Encoding UTF-8
LazyData true
RoxygenNote 7.1.2
Language en-US

R topics documented:

add_annotation .................................................. 3
add_attr .......................................................... 4
add_bar ............................................................ 5
add_class .......................................................... 6
add_line ............................................................ 6
add_point ........................................................... 7
add_tile ............................................................ 8
annot_to_list .................................................... 9
as_ComplexHeatmap ............................................. 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
ifelse2_pipe ...................................................... 19
ifelse_pipe ......................................................... 20
input_heatmap ..................................................... 21
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(),
  size = NULL,
  ...
)
```

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).
add_attr

**Description**

Add attribute to abject

**Usage**

```r
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**

```r
add_bar(.data, .column, palette = NULL, size = NULL, ...)
```

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

**Arguments**

- `.data` A ‘tbl_df’ formatted as | <ELEMENT> | <FEATURE> | <VALUE> | <...> |
- `.column` Vector of quotes
- `palette` A character vector of colors, or a function such as colorRamp2 (see examples).
- `size` A grid::unit object, e.g. unit(2, "cm"). This is the height or width of the annotation depending on the orientation.
- `...` The arguments that will be passed to top_annotation or left_annotation of the ComplexHeatmap container

**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_bar(inflection)
```
add_class  

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  

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

Usage
add_line(.data, .column, palette = NULL, size = NULL, ...)

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

Arguments
.data A 'tbl_df' formatted as | <ELEMENT> | <FEATURE> | <VALUE> | <...> |
.column Vector of quotes
palette A character vector of colors, or a function such as colorRamp2 (see examples).
size A grid::unit object, e.g. unit(2, "cm"). This is the height or width of the annotation depending on the orientation.
... The arguments that will be passed to top_annotation or left_annotation of the ComplexHeatmap container

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

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_line(inflection)

---

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

add_point(.data, .column, palette = NULL, size = NULL, ...)

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

Arguments

.data  
A ‘tbl_df’ formatted as | <ELEMENT> | <FEATURE> | <VALUE> | <...> |
.column  
A character vector of colors, or a function such as colorRamp2 (see examples).
.palette  
A grid::unit object, e.g. unit(2, "cm"). This is the height or width of the annotation depending on the orientation.
.size  
The arguments that will be passed to top_annotation or left_annotation of the ComplexHeatmap container

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(inflection)
```

---

**add_tile**

Adds a tile annotation layer to a `InputHeatmap`, that on evaluation creates a `ComplexHeatmap`

Description

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

Usage

```r
add_tile(.data, .column, palette = NULL, size = NULL, ...)
```

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

Arguments

- `.data`: A `tbl_df` formatted as `| <ELEMENT> | <FEATURE> | <VALUE> | <...> |`
- `.column`: Vector of quotes
- `palette`: A character vector of colors, or a function such as colorRamp2 (see examples).
- `size`: A grid::unit object, e.g. unit(2, "cm"). This is the height or width of the annotation depending on the orientation.
- `...`: The arguments that will be passed to top_annotation or left_annotation of the ComplexHeatmap container

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_tile(CAPRA_TOTAL)

hm %>% add_tile(inflection, palette = circlize::colorRamp2(c(0, 3, 10), c("white", "green"))
```

Description

`annot_to_list`

Usage

`annot_to_list(.data)`

Arguments

`.data` A data frame

Value

A list
as_ComplexHeatmap  

Creates a ‘ComplexHeatmap’ object for less standard plot manipulation (e.g. changing legend position)

Description

as_ComplexHeatmap() takes a ‘InputHeatmap’ object and produces a ‘Heatmap’ object

Usage

as_ComplexHeatmap(tidyHeatmap)

## S4 method for signature 'InputHeatmap'

as_ComplexHeatmap(tidyHeatmap)

Arguments

- **tidyHeatmap**  
  A ‘InputHeatmap’ object from tidyHeatmap::heatmap() call

Details

[Maturing]

Value

A ‘ComplexHeatmap’

Examples

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

---

as_matrix

Get matrix from tibble

Description

Get matrix from tibble

Usage

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

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
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**

```r
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**

```r
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**

```r
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**

```r
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, .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 = "none",
  palette_value = c("#440154FF", "#21908CFF", "#fedefa"),
  palette_grouping = list(),
  .scale = NULL,
  ...
)

heatmap_(
  .data,

## S4 method for signature 'tbl'

```r
heatmap(
  .data,
  .row, .column, .value,
  transform = NULL,
  scale = "none",
  palette_value = c("#440154FF", "#21908CFF", "#fefada"),
  palette_grouping = list(),
  .scale = NULL,
  ...
)
```

## S4 method for signature 'tbl_df'

```r
heatmap(
  .data,
  .row, .column, .value,
  transform = NULL,
  scale = "none",
  palette_value = c("#440154FF", "#21908CFF", "#fefada"),
  palette_grouping = list(),
  .scale = 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 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::magma(5))`
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", 
"#FDADC", "#CBD5E8")) or list(c("black", "red"))

.scale

DEPRECATED. please use scale instead \( \text{with no dot prefix} \).

... The arguments that will be passed to the Heatmap function of ComplexHeatmap backend

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' objects that gets evaluated to a 'ComplexHeatmap' object

A 'InputHeatmap' object

A 'InputHeatmap' object

A 'InputHeatmap' object

Examples

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

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
- \( .p_1 \) A boolean
- \( .p_2 \) ELSE IF condition
- \( .f_1 \) A function
- \( .f_2 \) A function
- \( .f_3 \) 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
- \( .f_1 \) A function
- \( .f_2 \) 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 = "none",
  palette_value = c("#440154FF", "#21908CFF", "#fefa0a"),
  palette_grouping = 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
- `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_down  
Adds a layers of symbols above the heatmap tiles to a 'InputHeatmap', that on evaluation creates a 'ComplexHeatmap'

Description

layer_arrow_down() from a 'InputHeatmap' object, adds a bar annotation layer.

Usage

layer_arrow_down(.data, ...)

## S4 method for signature 'InputHeatmap'
layer_arrow_down(.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.

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 %>% layer_arrow_down()
layer_arrow_up

Add 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.

Usage

layer_arrow_up(.data, ...)

## S4 method for signature 'InputHeatmap'
layer_arrow_up(.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.

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 %>% layer_arrow_up()
layer_diamond

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

Description

layer_diamond() from a 'InputHeatmap' object, adds a bar annotation layer.

Usage

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.

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 %>% layer_diamond()
layer_point

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

Description

layer_point() from a 'InputHeatmap' object, adds a bar annotation layer.

Usage

layer_point(.data, ...)

## S4 method for signature 'InputHeatmap'
layer_point(.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.

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 %>% layer_point()
layer_square

Description

layer_square() from a 'InputHeatmap' object, adds a bar annotation layer.

Usage

layer_square(.data, ...)

## S4 method for signature 'InputHeatmap'
layer_square(.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.

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 %>% layer_square()
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

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 8 columns.
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

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
quo_names(v)

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

Value
A character vector
save_pdf

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())
```
### Description

**save_pdf**

#### Usage

```r
## 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")

### Description

**save_pdf**

#### Usage

```r
## S4 method for signature 'InputHeatmap'
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")
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")

scale_design  Scale design matrix

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
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
split_rows(.data, number_of_groups)

## S4 method for signature ’InputHeatmap’
split_rows(.data, number_of_groups)

split_columns(.data, number_of_groups)

## 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.
wrap_heatmap

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'
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 %>% split_rows(2)

library(dplyr)

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

hm %>% split_columns(2)

---

wrap_heatmap

Wrap tidyHeatmap (ComplexHeatmap) in a patchwork-compliant patch

Description

In order to add tidyHeatmap (ComplexHeatmap) element to a patchwork they can be converted to a compliant representation using the 'wrap_heatmap()' function. This allows you to position either grobs, ggplot objects, patchwork objects, or even base graphics (if passed as a formula) in either the full area, the full plotting area (anything between and including the axis label), or the panel area (only the actual area where data is drawn).
wrap_heatmap

Usage

wrap_heatmap(
  panel = NULL,
  plot = NULL,
  full = NULL,
  clip = TRUE,
  ignore_tag = FALSE
)

## S4 method for signature 'InputHeatmap'
wrap_heatmap(
  panel = NULL,
  plot = NULL,
  full = NULL,
  clip = TRUE,
  ignore_tag = FALSE
)

Arguments

panel, plot, full
A grob, ggplot, patchwork, formula, raster, or nativeRaster object to add to the
respective area.

clip
Should the grobs be clipped if expanding outside its area

ignore_tag
Should tags be ignored for this patch. This is relevant when using automatic
tagging of plots and the content of the patch does not qualify for a tag.

Value

A wrapped_patch object

Examples

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

* datasets
  N52, 27
  pasilla, 27

add_annotation, 3
add_attr, 4
add_bar, 5
add_bar, InputHeatmap-method (add_bar), 5
add_class, 6
add_line, 6
add_line, InputHeatmap-method (add_line), 6
add_point, 7
add_point, InputHeatmap-method (add_point), 7
add_tile, 8
add_tile, InputHeatmap-method (add_tile), 8
annot_to_list, 9
as_ComplexHeatmap, 10
as_ComplexHeatmap, InputHeatmap-method (as_ComplexHeatmap), 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
ifelse2_pipe, 19
ifelse_pipe, 20
input_heatmap, 21

layer_arrow_down, 22
layer_arrow_down, InputHeatmap-method (layer_arrow_down), 22
layer_arrow_up, 23
layer_arrow_up, InputHeatmap-method (layer_arrow_up), 23
layer_diamond, 24
layer_diamond, InputHeatmap-method (layer_diamond), 24
layer_point, 25
layer_point, InputHeatmap-method (layer_point), 25
layer_square, 26
layer_square, InputHeatmap-method (layer_square), 26

N52, 27
parse_formula, 27
pasilla, 27
prepend, 28

quo_names, 28

save_pdf, 29
save_pdf, Heatmap-method, 30
save_pdf, InputHeatmap-method, 30
scale_design, 31
scale_robust, 31
select_closest_pairs, 32
split_columns (split_rows), 32
split_columns, InputHeatmap-method (split_rows), 32
split_rows, 32
split_rows, InputHeatmap-method (split_rows), 32

35
wrap_heatmap, 33
wrap_heatmap, InputHeatmap-method
   (wrap_heatmap), 33