

# Package ‘tidyHeatmap’

March 23, 2021

**Type** Package

**Title** A Tidy Implementation of Heatmap

**Version** 1.2.2

**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)

**Suggests** spelling,  
testthat,

vdiffr,  
 BiocManager,  
 knitr,  
 rmarkdown,  
 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
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 . . . . .	20
ifelse_pipe . . . . .	20
input_heatmap . . . . .	21
layer_symbol . . . . .	22
layer_symbol,InputHeatmap-method . . . . .	23
N52 . . . . .	23

parse\_formula . . . . . 24  
 pasilla . . . . . 24  
 prepend . . . . . 24  
 quo\_names . . . . . 25  
 save\_pdf . . . . . 25  
 save\_pdf,Heatmap-method . . . . . 26  
 save\_pdf,InputHeatmap-method . . . . . 27  
 scale\_design . . . . . 27  
 scale\_robust . . . . . 28  
 select\_closest\_pairs . . . . . 28

**Index** **29**

add\_annotation      *add\_annotation*

**Description**

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

**Usage**

```

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	<i>Add attribute to object</i>
----------	--------------------------------

---

**Description**

Add attribute to object

**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	<i>Adds a bar annotation layer to a ‘InputHeatmap’, that on evaluation creates a ‘ComplexHeatmap’</i>
---------	---

---

**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).

**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\_bar, InputHeatmap-method  
*add\_bar*

---

**Description**

add\_bar

**Usage**

```
## S4 method for signature 'InputHeatmap'
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).

**Value**

A 'add\_bar' object

---

add_class	<i>Add class to object</i>
-----------	----------------------------

---

**Description**

Add class to object

**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	<i>Adds a line annotation layer to a 'InputHeatmap', that on evaluation creates a 'ComplexHeatmap'</i>
----------	--

---

**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

```
library(dplyr)

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

hm %>% add_line()
```

---

add\_line, InputHeatmap-method  
*add\_line*

---

## Description

add\_line

## Usage

```
## 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 'add\_line' object

---

add_point	<i>Adds a point annotation layer to a 'InputHeatmap', that on evaluation creates a 'ComplexHeatmap'</i>
-----------	---

---

### Description

add\_point() from a 'InputHeatmap' object, adds a point annotation layer.

### Usage

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

## 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 'add\_point' object

### Examples

```
library(dplyr)

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

hm %>% add_point()
```



---

add_tile	<i>Adds a tile annotation layer to a 'InputHeatmap', that on evaluation creates a 'ComplexHeatmap'</i>
----------	--

---

## 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 'add\_tile' object

## 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	<i>annot_to_list</i>
---------------	----------------------

---

**Description**

annot\_to\_list

**Usage**

```
annot_to_list(.data)
```

**Arguments**

.data	A data frame
-------	--------------

**Value**

A list

---

as_matrix	<i>Get matrix from tibble</i>
-----------	-------------------------------

---

**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**

<code>.data</code>	A tibble of read counts
<code>list_input</code>	A list
<code>expected_type</code>	A character string

**Value**

A tbl

---

`drop_class`*Remove class to object*

---

**Description**

Remove class to object

**Usage**

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

**Arguments**

<code>var</code>	A tibble
<code>name</code>	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**

<code>x</code>	A numeric vector
<code>.abundance</code>	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**

<code>.data</code>	A tibble
<code>.abundance</code>	A character name of the abundance column

**Value**

A list of column enquo or error

---

get_elements	<i>Get column names either from user or from attributes</i>
--------------	---

---

**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	<i>Get column names either from user or from attributes</i>
-----------------------	---

---

**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**

<code>.data</code>	A tibble
<code>.sample</code>	A character name of the sample column
<code>.transcript</code>	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**

<code>.data</code>	A tibble
<code>.sample</code>	A character name of the sample column
<code>.transcript</code>	A character name of the transcript/gene column
<code>.abundance</code>	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**

<code>.data</code>	A 'tbl' formatted as  <SAMPLE> <TRANSCRIPT> <COUNT> <...>
<code>.column</code>	The name of the column horizontally presented in the heatmap
<code>.row</code>	The name of the column vertically presented in the heatmap
<code>.abundance</code>	The name of the transcript/gene abundance column

**Value**

A list

---

<code>heatmap</code>	<i>Creates a 'InputHeatmap' object from 'tbl_df' on evaluation creates a 'ComplexHeatmap'</i>
----------------------	---

---

**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(),
  ...
)
```

**Arguments**

<code>.data</code>	A 'tbl_df' formatted as   <ELEMENT>   <FEATURE>   <VALUE>   <...>
<code>.row</code>	The name of the column vertically presented in the heatmap
<code>.column</code>	The name of the column horizontally presented in the heatmap
<code>.value</code>	The name of the column for the value of the element/feature pair
<code>transform</code>	A function, used to transform <code>.value</code> row-wise (e.g., <code>transform = log1p</code> )
<code>.scale</code>	A character string. Possible values are <code>c("none", "row", "column", "both")</code>
<code>palette_value</code>	A character vector This is the palette that will be used as gradient for <code>.value</code> . For example <code>c("red", "white", "blue")</code> . For higher flexibility you can use <code>circize::colorRamp2(c(-2, -1, 0, 1, 2))</code> , <code>viridis::magma(5)</code>
<code>palette_grouping</code>	A list of character vectors. This is the list of palettes that will be used for grouping. For example <code>list(RColorBrewer::brewer.pal(8, "Accent"))</code> or <code>list(c("#B3E2CD", "#FDCDAC", "#CBD5E8"))</code> or <code>list(c("black", "red"))</code>
<code>annotation</code>	DEPRECATED. please use the annotation functions <code>add_*</code> function \(\(* one of tile, point, bar, line \).
<code>type</code>	DEPRECATED. please use the annotation functions <code>add_*</code> function \(\(* one of tile, point, bar, line \).
<code>palette_discrete</code>	DEPRECATED. please use the annotation functions <code>add_*</code> function \(\(* one of tile, point, bar, line \).
<code>palette_continuous</code>	DEPRECATED. please use the annotation functions <code>add_*</code> function \(\(* one of tile, point, bar, line \).
<code>...</code>	Further arguments to be passed to <code>ComplexHeatmap::Heatmap</code>

**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	<i>This is a generalisation of ifelse that accepts an object and return an objects</i>
--------------	--

---

### 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	<i>This is a generalisation of ifelse that accepts an object and return an objects</i>
-------------	--

---

### 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

---

input_heatmap	<i>input_heatmap</i>
---------------	----------------------

---

**Description**

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

**Usage**

```
input_heatmap(
  .data,
  .horizontal,
  .vertical,
  .abundance,
  transform = NULL,
  .scale = "row",
  palette_value = c("#440154FF", "#21908CFF", "#fefada"),
  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_symbol	<i>Adds a layers of symbols above the heatmap tiles to a ‘InputHeatmap’, that on evaluation creates a ‘ComplexHeatmap’</i>
--------------	--

---

### Description

layer\_symbol() from a ‘InputHeatmap’ object, adds a bar annotation layer.

### Usage

```
layer_symbol(.data, ..., symbol = NULL)
```

### 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.
symbol	NOT IMPLEMENTED YET

### 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_symbol()
```

---

```
layer_symbol, InputHeatmap-method
      layer_symbol
```

---

**Description**

layer\_symbol

**Usage**

```
## S4 method for signature 'InputHeatmap'
layer_symbol(.data, ..., symbol = NULL)
```

**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.
symbol	NOT IMPLEMENTED YET

**Value**

A 'layer\_symbol' object

---

N52	<i>Example data set N52</i>
-----	-----------------------------

---

**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 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	<i>Convert array of quosure (e.g. c(col_a, col_b)) into character vector</i>
-----------	--

---

**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	<i>Save plot on PDF file</i>
----------	------------------------------

---

**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  
*save\_pdf*

---

**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
      save_pdf
```

---

**Description**

save\_pdf

**Usage**

```
## 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      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

# Index

- \* **datasets**
  - N52, [23](#)
  - pasilla, [24](#)
- 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](#)
  
- ifelse\_pipe, [20](#)
- input\_heatmap, [21](#)
  
- layer\_symbol, [22](#)
- layer\_symbol, InputHeatmap-method,
  - [23](#)
  
- N52, [23](#)
  
- parse\_formula, [24](#)
- pasilla, [24](#)
- prepend, [24](#)
  
- quo\_names, [25](#)
  
- save\_pdf, [25](#)
- save\_pdf, Heatmap-method, [26](#)
- save\_pdf, InputHeatmap-method, [27](#)
- scale\_design, [27](#)
- scale\_robust, [28](#)
- select\_closest\_pairs, [28](#)