

# Package ‘ComplexHeatmap’

July 21, 2018

**Type** Package

**Title** Making Complex Heatmaps

**Version** 1.19.1

**Date** 2018-6-9

**Author** Zuguang Gu

**Maintainer** Zuguang Gu <z.gu@dkfz.de>

**Depends** R (>= 3.1.2), methods, grid, graphics, stats, grDevices

**Imports** circlize (>= 0.4.1), GetoptLong, colorspace, RColorBrewer, GlobalOptions (>= 0.1.0)

**Suggests** testthat (>= 0.3), knitr, markdown, cluster, MASS, pvclust, dendsort, HilbertCurve, Cairo, png, jpeg, tiff, fastcluster, dendextend (>= 1.0.1)

**VignetteBuilder** knitr

**Description** Complex heatmaps are efficient to visualize associations between different sources of data sets and reveal potential structures. Here the ComplexHeatmap package provides a highly flexible way to arrange multiple heatmaps and supports self-defined annotation graphics.

**biocViews** Software, Visualization, Sequencing

**URL** <https://github.com/jokergoo/ComplexHeatmap>

**License** MIT + file LICENSE

**git\_url** <https://git.bioconductor.org/packages/ComplexHeatmap>

**git\_branch** master

**git\_last\_commit** 8d50753

**git\_last\_commit\_date** 2018-06-26

**Date/Publication** 2018-07-21

## R topics documented:

ComplexHeatmap-package . . . . .	4
+AdditiveUnit . . . . .	5
AdditiveUnit . . . . .	6
AdditiveUnit-class . . . . .	6
add_heatmap-dispatch . . . . .	7

add_heatmap-Heatmap-method . . . . .	7
add_heatmap-HeatmapAnnotation-method . . . . .	8
add_heatmap-HeatmapList-method . . . . .	9
adjust_dend_by_leaf_width . . . . .	10
annotation_legend_size-HeatmapList-method . . . . .	11
anno_barplot . . . . .	12
anno_boxplot . . . . .	13
anno_density . . . . .	14
anno_histogram . . . . .	15
anno_link . . . . .	16
anno_oncoprint_barplot . . . . .	17
anno_points . . . . .	17
anno_text . . . . .	18
ColorMapping . . . . .	19
ColorMapping-class . . . . .	20
color_mapping_legend-ColorMapping-method . . . . .	21
columnAnnotation . . . . .	23
column_anno_barplot . . . . .	23
column_anno_boxplot . . . . .	24
column_anno_density . . . . .	25
column_anno_histogram . . . . .	25
column_anno_link . . . . .	26
column_anno_points . . . . .	27
column_anno_text . . . . .	27
column_dend-dispatch . . . . .	28
column_dend-Heatmap-method . . . . .	28
column_dend-HeatmapList-method . . . . .	29
column_order-dispatch . . . . .	30
column_order-Heatmap-method . . . . .	30
column_order-HeatmapList-method . . . . .	31
component_height-dispatch . . . . .	31
component_height-Heatmap-method . . . . .	32
component_height-HeatmapList-method . . . . .	33
component_width-dispatch . . . . .	33
component_width-Heatmap-method . . . . .	34
component_width-HeatmapList-method . . . . .	35
decorate_annotation . . . . .	35
decorate_column_dend . . . . .	36
decorate_column_names . . . . .	37
decorate_column_title . . . . .	38
decorate_dend . . . . .	38
decorate_dimnames . . . . .	39
decorate_heatmap_body . . . . .	40
decorate_row_dend . . . . .	41
decorate_row_names . . . . .	42
decorate_row_title . . . . .	43
decorate_title . . . . .	43
densityHeatmap . . . . .	44
dist2 . . . . .	46
draw-dispatch . . . . .	47
draw-Heatmap-method . . . . .	48
draw-HeatmapAnnotation-method . . . . .	49

draw-HeatmapList-method . . . . .	50
draw-SingleAnnotation-method . . . . .	52
draw_annotation-Heatmap-method . . . . .	53
draw_annotation_legend-HeatmapList-method . . . . .	54
draw_dend-Heatmap-method . . . . .	54
draw_dimnames-Heatmap-method . . . . .	55
draw_heatmap_body-Heatmap-method . . . . .	56
draw_heatmap_legend-HeatmapList-method . . . . .	57
draw_heatmap_list-HeatmapList-method . . . . .	58
draw_title-dispatch . . . . .	59
draw_title-Heatmap-method . . . . .	59
draw_title-HeatmapList-method . . . . .	60
enhanced_basicplot . . . . .	61
get_color_mapping_list-HeatmapAnnotation-method . . . . .	62
get_color_mapping_param_list-HeatmapAnnotation-method . . . . .	63
grid.dendrogram . . . . .	63
grid.dendrogram2 . . . . .	64
Heatmap . . . . .	65
Heatmap-class . . . . .	71
HeatmapAnnotation . . . . .	73
HeatmapAnnotation-class . . . . .	75
HeatmapList . . . . .	75
HeatmapList-class . . . . .	76
heatmap_legend_size-HeatmapList-method . . . . .	78
ht_global_opt . . . . .	78
is_abs_unit . . . . .	80
Legend . . . . .	81
make_column_cluster-Heatmap-method . . . . .	82
make_layout-dispatch . . . . .	83
make_layout-Heatmap-method . . . . .	84
make_layout-HeatmapList-method . . . . .	84
make_row_cluster-Heatmap-method . . . . .	87
map_to_colors-ColorMapping-method . . . . .	88
max_text_height . . . . .	89
max_text_width . . . . .	89
oncoPrint . . . . .	90
packLegend . . . . .	93
plotDataFrame . . . . .	94
prepare-Heatmap-method . . . . .	95
rowAnnotation . . . . .	96
row_anno_barplot . . . . .	97
row_anno_boxplot . . . . .	97
row_anno_density . . . . .	98
row_anno_histogram . . . . .	99
row_anno_link . . . . .	99
row_anno_points . . . . .	100
row_anno_text . . . . .	101
row_dend-dispatch . . . . .	101
row_dend-Heatmap-method . . . . .	102
row_dend-HeatmapList-method . . . . .	102
row_order-dispatch . . . . .	103
row_order-Heatmap-method . . . . .	103

row_order-HeatmapList-method . . . . .	104
selectArea . . . . .	105
set_component_height-Heatmap-method . . . . .	105
show-ColorMapping-method . . . . .	106
show-dispatch . . . . .	107
show-Heatmap-method . . . . .	107
show-HeatmapAnnotation-method . . . . .	108
show-HeatmapList-method . . . . .	109
show-SingleAnnotation-method . . . . .	109
SingleAnnotation . . . . .	110
SingleAnnotation-class . . . . .	112
unify_mat_list . . . . .	113

## Index 114

---

ComplexHeatmap-package

*Making complex heatmap*

---

### Description

Making complex heatmap

### Details

This package aims to provide a simple and flexible way to arrange multiple heatmaps as well as self-defining annotation graphics.

The package is implemented in an object-oriented way. Components of heatmap lists are abstracted into several classes.

- [Heatmap-class](#): a single heatmap containing heatmap body, row/column names, titles, dendrograms and column annotations.
- [HeatmapList-class](#): a list of heatmaps and row annotations.
- [HeatmapAnnotation-class](#): a list of row annotations or column annotations.

There are also several internal classes:

- [SingleAnnotation-class](#): a single row annotation or column annotation.
- [ColorMapping-class](#): mapping from values to colors.

For plotting one single heatmap, please go to the documentation page of [Heatmap](#). For plotting multiple heatmaps, please go to [HeatmapList-class](#) and `+.AdditiveUnit`.

The vignette provides detailed explanation of how to use this package.

### Examples

```
# There is no example
NULL
```

---

+.AdditiveUnit                    *Add heatmaps or row annotations to a heatmap list*

---

## Description

Add heatmaps or row annotations to a heatmap list

## Usage

```
## S3 method for class 'AdditiveUnit'  
x + y
```

## Arguments

x                    a [Heatmap-class](#) object, a [HeatmapAnnotation-class](#) object or a [HeatmapList-class](#) object.

y                    a [Heatmap-class](#) object, a [HeatmapAnnotation-class](#) object or a [HeatmapList-class](#) object.

## Details

It is only a helper function. It actually calls [add\\_heatmap, Heatmap-method](#), [add\\_heatmap, HeatmapList-method](#) or [add\\_heatmap, HeatmapAnnotation-method](#) depending on the class of the input objects.

The [HeatmapAnnotation-class](#) object to be added should only be row annotations.

## Value

A [HeatmapList-class](#) object.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
mat = matrix(rnorm(80, 2), 8, 10)  
mat = rbind(mat, matrix(rnorm(40, -2), 4, 10))  
rownames(mat) = letters[1:12]  
colnames(mat) = letters[1:10]  
  
ht = Heatmap(mat)  
ht + ht  
ht + ht + ht  
  
ht_list = ht + ht  
ht + ht_list  
  
ha = rowAnnotation(points = row_anno_points(1:12))  
ht + ha  
ht_list + ha  
  
ha + ha + ht
```

---

AdditiveUnit	<i>Constructor method for AdditiveUnit class</i>
--------------	--

---

**Description**

Constructor method for AdditiveUnit class

**Usage**

```
AdditiveUnit(...)
```

**Arguments**

...                    black hole arguments.

**Details**

This method is not used in the package.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this function  
NULL
```

---

AdditiveUnit-class	<i>An internal class</i>
--------------------	--------------------------

---

**Description**

An internal class

**Details**

This class is a super class for [Heatmap-class](#), [HeatmapList-class](#) and [HeatmapAnnotation-class](#) classes. It is only designed for + generic method so that above three classes can be appended to each other.

**Examples**

```
# no example  
NULL
```

---

add\_heatmap-dispatch *Method dispatch page for add\_heatmap*

---

### Description

Method dispatch page for add\_heatmap.

### Dispatch

add\_heatmap can be dispatched on following classes:

- [add\\_heatmap, HeatmapAnnotation-method, HeatmapAnnotation-class](#) class method
- [add\\_heatmap, HeatmapList-method, HeatmapList-class](#) class method
- [add\\_heatmap, Heatmap-method, Heatmap-class](#) class method

### Examples

```
# no example
NULL
```

---

add\_heatmap-Heatmap-method

*Add heatmaps or row annotations as a heatmap list*

---

### Description

Add heatmaps or row annotations as a heatmap list

### Usage

```
## S4 method for signature 'Heatmap'
add_heatmap(object, x)
```

### Arguments

object	a <a href="#">Heatmap-class</a> object.
x	a <a href="#">Heatmap-class</a> object, a <a href="#">HeatmapAnnotation-class</a> object or a <a href="#">HeatmapList-class</a> object.

### Details

There is a shortcut function `+.AdditiveUnit`.

### Value

A [HeatmapList-class](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
mat = matrix(rnorm(80, 2), 8, 10)
mat = rbind(mat, matrix(rnorm(40, -2), 4, 10))
rownames(mat) = letters[1:12]
colnames(mat) = letters[1:10]

ht = Heatmap(mat)
add_heatmap(ht, ht)

ha = HeatmapAnnotation(points = anno_points(1:12, which = "row"),
  which = "row")
add_heatmap(ht, ha)
```

---

add\_heatmap-HeatmapAnnotation-method

*Add row annotations or heatmaps as a heatmap list*

---

**Description**

Add row annotations or heatmaps as a heatmap list

**Usage**

```
## S4 method for signature 'HeatmapAnnotation'
add_heatmap(object, x)
```

**Arguments**

object	a <a href="#">HeatmapAnnotation-class</a> object.
x	a <a href="#">Heatmap-class</a> object, a <a href="#">HeatmapAnnotation-class</a> object or a <a href="#">HeatmapList-class</a> object.

**Details**

There is a shortcut function `+.AdditiveUnit`.

**Value**

A [HeatmapList-class](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>



**Examples**

```
mat = matrix(rnorm(80, 2), 8, 10)
mat = rbind(mat, matrix(rnorm(40, -2), 4, 10))
rownames(mat) = letters[1:12]
colnames(mat) = letters[1:10]

ht = Heatmap(mat)

ha = HeatmapAnnotation(points = anno_points(1:12, which = "row"),
  which = "row")
add_heatmap(ha, ht)
```

---

add\_heatmap-HeatmapList-method

*Add heatmaps and row annotations to the heatmap list*

---

**Description**

Add heatmaps and row annotations to the heatmap list

**Usage**

```
## S4 method for signature 'HeatmapList'
add_heatmap(object, x)
```

**Arguments**

object	a <a href="#">HeatmapList-class</a> object.
x	a <a href="#">Heatmap-class</a> object or a <a href="#">HeatmapAnnotation-class</a> object or a <a href="#">HeatmapList-class</a> object.

**Details**

There is a shortcut function `+.AdditiveUnit`.

**Value**

A [HeatmapList-class](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
mat = matrix(rnorm(80, 2), 8, 10)
mat = rbind(mat, matrix(rnorm(40, -2), 4, 10))
rownames(mat) = letters[1:12]
colnames(mat) = letters[1:10]

ht = Heatmap(mat)
ht_list = ht + ht
```

```
add_heatmap(ht_list, ht)

ha = HeatmapAnnotation(points = anno_points(1:12, which = "row"),
  which = "row")
add_heatmap(ht_list, ha)
```

---

adjust\_dend\_by\_leaf\_width

*Adjust dendrogram based on width of leaves*

---

## Description

Adjust dendrogram based on width of leaves

## Usage

```
adjust_dend_by_leaf_width(dend, width = 1, offset = 0)
```

## Arguments

dend	a <a href="#">dendrogram</a> object.
width	a vector of width. The order of width SHOULD be same as the order of original elements before clustering.
offset	offset to $x = 0$

## Details

In the standard [dendrogram](#) object, leaves locate at  $x = 0.5, 1.5, \dots, n - 0.5$ , which means, the width of leaves are always 1 and the distance to neighbouring leaves is always 1 as well. Here [adjust\\_dend\\_by\\_leaf\\_width](#) adjusts the dendrogram by setting different width for leaves so that leaves have unequal distance to other leaves.

The adjusted dendrogram can be sent to [grid.dendrogram2](#) to make the dendrogram.

For each branch as well each leaf, a new attribute of  $x$  is added which is the position of the middle point or the leaf. For each leaf, a new attribute of  $width$  is added which is the width of current leaf.

## Value

A [dendrogram](#) object. The adjustment will not affect other standard dendrogram functions.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
m = matrix(rnorm(100), 10)
dend = as.dendrogram(hclust(dist(m)))
dend = adjust_dend_by_leaf_width(dend, width = 1:10)
require(dendextend)
get_leaves_attr(dend, "label")
get_leaves_attr(dend, "width")
get_leaves_attr(dend, "x")
```

---

annotation\_legend\_size-HeatmapList-method  
*Size of the annotation legend viewport*

---

## Description

Size of the annotation legend viewport

## Usage

```
## S4 method for signature 'HeatmapList'  
annotation_legend_size(object, legend_list = list(), ...)
```

## Arguments

`object` a [HeatmapList-class](#) object.  
`legend_list` a list of self-defined legend, should be wrapped into [grob](#) objects.  
`...` graphic parameters passed to [color\\_mapping\\_legend, ColorMapping-method](#).

## Details

Legends for all heatmaps or legends for all annotations will be put in one viewport. This function calculates the size of such viewport. Note graphic parameters for legends will affect the size.

This function is only for internal use.

## Value

A [unit](#) object.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
# no example for this internal method  
NULL
```

anno\_barplot

*Using barplot as annotation***Description**

Using barplot as annotation

**Usage**

```
anno_barplot(x, baseline = "min", which = c("column", "row"), border = TRUE, bar_width = 0.6,
             gp = gpar(fill = "#CCCCCC"), ylim = NULL, axis = FALSE, axis_side = NULL,
             axis_gp = gpar(fontsize = 8), axis_direction = c("normal", "reverse"), ...)
```

**Arguments**

x	a vector of numeric values. If the value is a matrix, columns of the matrix will be represented as stacked barplots. Note for stacked barplots, each row in the matrix should only contain values with same sign (either all positive or all negative).
baseline	baseline for bars. The value should be "min" or "max", or a numeric value. It is enforced to be zero for stacked barplots.
which	is the annotation a column annotation or a row annotation?
border	whether show border of the annotation component
bar_width	relative width of the bars, should less than one
gp	graphic parameters. If it is the stacked barplots, the length of the graphic parameter should be same as the number of stacks.
ylim	data ranges.
axis	whether add axis
axis_side	if it is placed as column annotation, value can only be "left" or "right". If it is placed as row annotation, value can only be "bottom" or "top".
axis_gp	graphic parameters for axis
axis_direction	if the annotation is row annotation, should the axis be from left to right (default) or follow the reversed direction?
...	for future use.

**Value**A graphic function which can be set in [HeatmapAnnotation](#) constructor method.**Author(s)**

Zuguang Gu &lt;z.gu@dkfz.de&gt;

**Examples**

```
f = anno_barplot(rnorm(10))
grid.newpage(); f(1:10)

f = anno_barplot(rnorm(10), which = "row")
grid.newpage(); f(1:10)
```

---

anno_boxplot	<i>Using boxplot as annotation</i>
--------------	------------------------------------

---

## Description

Using boxplot as annotation

## Usage

```
anno_boxplot(x, which = c("column", "row"), border = TRUE,  
             gp = gpar(fill = "#CCCCCC"), ylim = NULL, outline = TRUE,  
             pch = 16, size = unit(2, "mm"), axis = FALSE, axis_side = NULL,  
             axis_gp = gpar(fontsize = 8), axis_direction = c("normal", "reverse"))
```

## Arguments

x	a matrix or a list. If x is a matrix and if which is column, statistics for boxplot is calculated by columns, if which is row, the calculation is by rows.
which	is the annotation a column annotation or a row annotation?
border	whether show border of the annotation component
gp	graphic parameters
ylim	data ranges.
outline	whether draw outliers
pch	point type
size	point size
axis	whether add axis
axis_side	if it is placed as column annotation, value can only be "left" or "right". If it is placed as row annotation, value can only be "bottom" or "top".
axis_gp	graphic parameters for axis
axis_direction	if the annotation is row annotation, should the axis be from left to right (default) or follow the reversed direction?

## Value

A graphic function which can be set in [HeatmapAnnotation](#) constructor method.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
mat = matrix(rnorm(32), nrow = 4)  
f = anno_boxplot(mat)  
grid.newpage(); f(1:8)  
  
f = anno_boxplot(mat, which = "row")  
grid.newpage(); f(1:4)
```

```
lt = lapply(1:4, function(i) rnorm(8))
f = anno_boxplot(lt)
grid.newpage(); f(1:4)
```

---

anno\_density

*Using kernel density as annotation*

---

## Description

Using kernel density as annotation

## Usage

```
anno_density(x, which = c("column", "row"), gp = gpar(fill = "#CCCCCC"),
             type = c("lines", "violin", "heatmap"), ...)
```

## Arguments

x	a matrix or a list. If x is a matrix and if which is column, statistics for density is calculated by columns, if which is row, the calculation is by rows.
which	is the annotation a column annotation or a row annotation?
gp	graphic parameters. Note it is ignored if type equals to heatmap.
type	which type of graphics is used to represent density distribution.
...	pass to <a href="#">density</a>

## Value

A graphic function which can be set in [HeatmapAnnotation](#) constructor method.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
mat = matrix(rnorm(32), nrow = 4)
f = anno_density(mat)
grid.newpage(); f(1:8)

f = anno_density(mat, which = "row", type = "violin")
grid.newpage(); f(1:4)

lt = lapply(1:4, function(i) rnorm(8))
f = anno_density(lt, type = "heatmap")
grid.newpage(); f(1:4)
```

---

anno_histogram	<i>Using histogram as annotation</i>
----------------	--------------------------------------

---

## Description

Using histogram as annotation

## Usage

```
anno_histogram(x, which = c("column", "row"), gp = gpar(fill = "#CCCCCC"), ...)
```

## Arguments

x	a matrix or a list. If x is a matrix and if which is column, statistics for histogram is calculated by columns, if which is row, the calculation is by rows.
which	is the annotation a column annotation or a row annotation?
gp	graphic parameters
...	pass to <a href="#">hist</a>

## Value

A graphic function which can be set in [HeatmapAnnotation](#) constructor method.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
mat = matrix(rnorm(32), nrow = 4)
f = anno_histogram(mat)
grid.newpage(); f(1:8)

f = anno_histogram(mat, which = "row")
grid.newpage(); f(1:4)

lt = lapply(1:4, function(i) rnorm(8))
f = anno_histogram(lt)
grid.newpage(); f(1:4)
```

---

anno\_link *Link annotation with labels*

---

### Description

Link annotation with labels

### Usage

```
anno_link(at, labels, which = c("column", "row"), side = ifelse(which == "column", "top", "right")
          lines_gp = gpar(), labels_gp = gpar(), padding = 0.25, link_width = NULL, extend = 0)
```

### Arguments

at	numeric index in the original matrix
labels	corresponding labels
which	column annotation or row annotation
side	side of the labels. If it is a column annotation, permitted values are "top" and "bottom"; If it is a row annotation, permitted values are "left" and "right".
lines_gp	graphic settings for the segments
labels_gp	graphic settings for the labels
padding	padding between labels if they are attached to each other
link_width,	width of the segments.
extend	by default, the region for the labels has the same width (if it is a column annotation) or same height (if it is a row annotation) as the heatmap. The size can be extended by this options. The value can be a proportion number or a <a href="#">unit</a> object. The length can be either one or two.

### Details

Sometimes there are many rows or columns in the heatmap and we want to mark some of the rows. This annotation function is used to mark these rows and connect labels and corresponding rows with links.

### Value

A graphic function which can be set in [HeatmapAnnotation](#) constructor method.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
mat = matrix(rnorm(10000), nr = 1000)
labels = sample(letters, 20, replace = TRUE)
Heatmap(mat, show_row_dend = FALSE, show_column_dend = FALSE) +
rowAnnotation(link = row_anno_link(at = sample(1000, 20), labels = labels),
              width = unit(1, "cm") + max_text_width(labels))
```



---

 anno\_ancoprint\_barplot

*Column barplot annotation for oncoPrint*


---

**Description**

Column barplot annotation for oncoPrint

**Usage**

```
anno_ancoprint_barplot()
```

**Details**

This function is only used for column annotation

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example
NULL
```

---

 anno\_points

*Using points as annotation*


---

**Description**

Using points as annotation

**Usage**

```
anno_points(x, which = c("column", "row"), border = TRUE, gp = gpar(), pch = 16,
  size = unit(2, "mm"), ylim = NULL, axis = FALSE, axis_side = NULL,
  axis_gp = gpar(fontsize = 8), axis_direction = c("normal", "reverse"), ...)
```

**Arguments**

x	a vector of numeric values.
which	is the annotation a column annotation or a row annotation?
border	whether show border of the annotation component
gp	graphic parameters.
pch	point type.
size	point size.
ylim	data ranges.

axis	whether add axis.
axis_side	if it is placed as column annotation, value can only be "left" or "right". If it is placed as row annotation, value can only be "bottom" or "top".
axis_gp	graphic parameters for axis
axis_direction	if the annotation is row annotation, should the axis be from left to right (default) or follow the reversed direction?
...	for future use.

**Value**

A graphic function which can be set in [HeatmapAnnotation](#) constructor method.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
f = anno_points(rnorm(10))
grid.newpage(); f(1:10)
```

---

anno\_text

*Using text as annotation*


---

**Description**

Using text as annotation

**Usage**

```
anno_text(x, which = c("column", "row"), gp = gpar(), rot = 0,
just = NULL, offset = unit(0.5, "npc"))
```

**Arguments**

x	a vector of text
which	is the annotation a column annotation or a row annotation?
gp	graphic parameters.
rot	rotation of text
just	justification of text, pass to <a href="#">grid.text</a>
offset	if it is a row annotation, offset corresponds to the x-coordinates of text. and if it is a column annotation, offset corresponds to the y-coordinates of text. The value should be a <a href="#">unit</a> object.

**Value**

A graphic function which can be set in [HeatmapAnnotation](#) constructor method.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
mat = matrix(rnorm(100), 10)
colnames(mat) = letters[1:10]
rownames(mat) = LETTERS[1:10]
long_cn = do.call("paste0", rep(list(colnames(mat)), 4)) # just to construct long text
ha_rot_cn = HeatmapAnnotation(text = anno_text(long_cn, rot = 45, offset = unit(5, "mm")))
Heatmap(mat, name = "foo", top_annotation = ha_rot_cn, top_annotation_height = unit(1.2, "cm"))
```

---

ColorMapping

*Constructor methods for ColorMapping class*

---

**Description**

Constructor methods for ColorMapping class

**Usage**

```
ColorMapping(name, colors = NULL, levels = NULL,
             col_fun = NULL, breaks = NULL, na_col = "#FFFFFF")
```

**Arguments**

name	name for this color mapping. The name is automatically generated if it is not specified.
colors	discrete colors.
levels	levels that correspond to colors. If colors is name indexed, levels can be ignored.
col_fun	color mapping function that maps continuous values to colors.
breaks	breaks for the continuous color mapping. If col_fun is generated by <a href="#">colorRamp2</a> , breaks can be ignored.
na_col	colors for NA values.

**Details**

colors and levels are used for discrete color mapping, col\_fun and breaks are used for continuous color mapping.

**Value**

A [ColorMapping-class](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
# discrete color mapping for characters
cm = ColorMapping(name = "test",
  colors = c("blue", "white", "red"),
  levels = c("a", "b", "c"))
cm

# discrete color mapping for numeric values
cm = ColorMapping(name = "test",
  colors = c("blue", "white", "red"),
  levels = c(1, 2, 3))
cm

# continuous color mapping
require(circlize)
cm = ColorMapping(name = "test",
  col_fun = colorRamp2(c(0, 0.5, 1), c("blue", "white", "red")))
cm
```

---

ColorMapping-class      *Class to map values to colors*

---

## Description

Class to map values to colors

## Details

The [ColorMapping-class](#) handles color mapping with both discrete values and continuous values. Discrete values are mapped by setting a vector of colors and continuous values are mapped by setting a color mapping function.

## Methods

The [ColorMapping-class](#) provides following methods:

- [ColorMapping](#): constructor methods.
- [map\\_to\\_colors, ColorMapping-method](#): mapping values to colors.
- [color\\_mapping\\_legend, ColorMapping-method](#): draw legend or get legend as a [grob](#) object.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
# for examples, please go to `ColorMapping` method page
NULL
```

---

color\_mapping\_legend-ColorMapping-method  
*Draw legend based on color mapping*

---

## Description

Draw legend based on color mapping

## Usage

```
## S4 method for signature 'ColorMapping'
color_mapping_legend(object, ...,
  plot = TRUE,
  title = object@name,
  title_gp = gpar(fontsize = 10, fontface = "bold"),
  title_position = c("topleft", "topcenter", "leftcenter", "lefttop"),
  color_bar = object@type,
  grid_height = unit(4, "mm"),
  grid_width = unit(4, "mm"),
  border = NULL,
  at = object@levels,
  labels = at,
  labels_gp = gpar(fontsize = 10),
  nrow = NULL,
  ncol = 1,
  by_row = FALSE,
  legend_height = NULL, legend_width = NULL,
  legend_direction = c("vertical", "horizontal"),
  param = NULL)
```

## Arguments

object	a <a href="#">ColorMapping-class</a> object.
plot	whether to plot or just return the size of the legend viewport.
title	title of the legend, by default it is the name of the legend
title_gp	graphical parameters for legend title
title_position	position of the title
color_bar	a string of "continuous" or "discrete". If the mapping is continuous, whether show the legend as discrete color bar or continuous color bar
grid_height	height of each legend grid.
grid_width	width of each legend grid.
border	color for legend grid borders.
at	break values of the legend
labels	labels corresponding to break values
labels_gp	graphical parameters for legend labels
nrow	if there are too many legend grids, they can be put as an array, this controls number of rows

ncol	if there are too many legend grids, they can be put as an array, this controls number of columns
by_row	when there are multiple columns for legends, whether to arrange them by rows.
legend_height	height of the legend, only works when color_bar is continuous and direction is vertical
legend_width	width of the legend, only works when color_bar is continuous and direction is horizontal
legend_direction	when color_bar is continuous, should the legend be vertical or horizontal? When color_bar is discrete, should the items in the legend proceed vertically or horizontally?
param	will be parsed if the parameters are specified as a list
...	pass to <a href="#">viewport</a> .

### Details

A viewport is created which contains a legend title, legend grids and corresponding labels.

This function will be improved in the future to support more types of legends.

### Value

A [grob](#) object which contains the legend

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
# discrete color mapping for characters
cm = ColorMapping(name = "test",
  colors = c("blue", "white", "red"),
  levels = c("a", "b", "c"))
grid.newpage()
color_mapping_legend(cm)

# discrete color mapping for numeric values
cm = ColorMapping(name = "test",
  colors = c("blue", "white", "red"),
  levels = c(1, 2, 3))
grid.newpage()
color_mapping_legend(cm)

# continuous color mapping
require(circlize)
cm = ColorMapping(name = "test",
  col_fun = colorRamp2(c(0, 0.5, 1), c("blue", "white", "red")))
grid.newpage()
color_mapping_legend(cm, title_gp = gpar(fontsize = 16))
```

---

columnAnnotation	<i>Construct column annotations</i>
------------------	-------------------------------------

---

**Description**

Construct column annotations

**Usage**

```
columnAnnotation(...)
```

**Arguments**

... pass to [HeatmapAnnotation](#)

**Details**

The function is identical to

```
HeatmapAnnotation(..., which = "column")
```

**Value**

A [HeatmapAnnotation-class](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
df = data.frame(type = c("a", "a", "a", "b", "b", "b"))
ha = rowAnnotation(df = df)
```

---

column_anno_barplot	<i>Column annotation which is represented as barplots</i>
---------------------	---

---

**Description**

Column annotation which is represented as barplots

**Usage**

```
column_anno_barplot(...)
```

**Arguments**

... pass to [anno\\_barplot](#)

**Details**

A wrapper of [anno\\_barplot](#) with pre-defined which to column.

**Value**

See help page of [anno\\_barplot](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example  
NULL
```

---

column\_anno\_boxplot    *Column annotation which is represented as boxplots*

---

**Description**

Column annotation which is represented as boxplots

**Usage**

```
column_anno_boxplot(...)
```

**Arguments**

```
...                    pass to anno\_boxplot
```

**Details**

A wrapper of [anno\\_boxplot](#) with pre-defined which to column.

**Value**

See help page of [anno\\_boxplot](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example  
NULL
```



---

column\_anno\_density *Column annotation which is represented as density plot*

---

**Description**

Column annotation which is represented as density plot

**Usage**

```
column_anno_density(...)
```

**Arguments**

... pass to [anno\\_density](#)

**Details**

A wrapper of [anno\\_density](#) with pre-defined which to column.

**Value**

See help page of [anno\\_density](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example  
NULL
```

---

column\_anno\_histogram *Column annotation which is represented as histogram*

---

**Description**

Column annotation which is represented as histogram

**Usage**

```
column_anno_histogram(...)
```

**Arguments**

... pass to [anno\\_histogram](#)

**Details**

A wrapper of [anno\\_histogram](#) with pre-defined which to column.

**Value**

See help page of [anno\\_histogram](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example  
NULL
```

---

column_anno_link	<i>Column annotation which is represented as links</i>
------------------	--

---

**Description**

Column annotation which is represented as links

**Usage**

```
column_anno_link(...)
```

**Arguments**

```
...          pass to anno\_link
```

**Details**

A wrapper of [anno\\_link](#) with pre-defined which to column.

**Value**

See help page of [anno\\_link](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example  
NULL
```

---

column_anno_points	<i>Column annotation which is represented as points</i>
--------------------	---

---

**Description**

Column annotation which is represented as points

**Usage**

```
column_anno_points(...)
```

**Arguments**

... pass to [anno\\_points](#)

**Details**

A wrapper of [anno\\_points](#) with pre-defined which to column.

**Value**

See help page of [anno\\_points](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example  
NULL
```

---

column_anno_text	<i>Column annotation which is represented as text</i>
------------------	---

---

**Description**

Column annotation which is represented as text

**Usage**

```
column_anno_text(...)
```

**Arguments**

... pass to [anno\\_text](#)

**Details**

A wrapper of [anno\\_text](#) with pre-defined which to column.

**Value**

See help page of [anno\\_text](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example
NULL
```

---

column\_dend-dispatch *Method dispatch page for column\_dend*

---

**Description**

Method dispatch page for column\_dend.

**Dispatch**

column\_dend can be dispatched on following classes:

- [column\\_dend,HeatmapList-method](#), [HeatmapList-class](#) class method
- [column\\_dend,Heatmap-method](#), [Heatmap-class](#) class method

**Examples**

```
# no example
NULL
```

---

column\_dend-Heatmap-method  
*Get column dendrograms from a heatmap*

---

**Description**

Get column dendrograms from a heatmap

**Usage**

```
## S4 method for signature 'Heatmap'
column_dend(object)
```

**Arguments**

object            a [Heatmap-class](#) object

**Value**

A dendrogram object

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
mat = matrix(rnorm(100), 10)
ht = Heatmap(mat)
column_dend(ht)
ht = Heatmap(mat, km = 2)
column_dend(ht)
```

---

column\_dend-HeatmapList-method

*Get column dendrograms from a heatmap list*

---

**Description**

Get column dendrograms from a heatmap list

**Usage**

```
## S4 method for signature 'HeatmapList'
column_dend(object)
```

**Arguments**

object            a [HeatmapList-class](#) object

**Value**

A list of dendrograms for which dendrogram corresponds to each matrix

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
mat = matrix(rnorm(100), 10)
ht_list = Heatmap(mat) + Heatmap(mat)
column_dend(ht_list)
ht_list = Heatmap(mat, km = 2) + Heatmap(mat)
column_dend(ht_list)
```

column\_order-dispatch *Method dispatch page for column\_order*

---

### Description

Method dispatch page for column\_order.

### Dispatch

column\_order can be dispatched on following classes:

- [column\\_order, HeatmapList-method, HeatmapList-class](#) class method
- [column\\_order, Heatmap-method, Heatmap-class](#) class method

### Examples

```
# no example
NULL
```

---

column\_order-Heatmap-method

*Get column order from a heatmap list*

---

### Description

Get column order from a heatmap list

### Usage

```
## S4 method for signature 'Heatmap'
column_order(object)
```

### Arguments

object            a [Heatmap-class](#) object

### Value

A vector containing column orders

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
mat = matrix(rnorm(100), 10)
ht = Heatmap(mat)
column_order(ht)
ht = Heatmap(mat, km = 2)
column_order(ht)
```

---

column\_order-HeatmapList-method

*Get column order from a heatmap list*

---

### Description

Get column order from a heatmap list

### Usage

```
## S4 method for signature 'HeatmapList'  
column_order(object)
```

### Arguments

object            a [HeatmapList-class](#) object

### Value

A list contains column orders which correspond every matrix

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
mat = matrix(rnorm(100), 10)  
ht_list = Heatmap(mat) + Heatmap(mat)  
column_order(ht_list)  
ht = Heatmap(mat, km = 2) + Heatmap(mat)  
column_order(ht_list)
```

---

component\_height-dispatch

*Method dispatch page for component\_height*

---

### Description

Method dispatch page for component\_height.

### Dispatch

component\_height can be dispatched on following classes:

- [component\\_height,HeatmapList-method, HeatmapList-class](#) class method
- [component\\_height,Heatmap-method, Heatmap-class](#) class method

**Examples**

```
# no example  
NULL
```

---

```
component_height-Heatmap-method  
Height of each heatmap component
```

---

**Description**

Height of each heatmap component

**Usage**

```
## S4 method for signature 'Heatmap'  
component_height(object, k = 1:9)
```

**Arguments**

object            a [Heatmap-class](#) object.  
k                which component in the heatmap, see [Heatmap-class](#).

**Details**

This function is only for internal use.

**Value**

A [unit](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this internal method
```



---

component\_height-HeatmapList-method

*Height of each heatmap list component*

---

### Description

Height of each heatmap list component

### Usage

```
## S4 method for signature 'HeatmapList'  
component_height(object, k = 1:7)
```

### Arguments

object            a [HeatmapList-class](#) object.  
k                which component in the heatmap list, see [HeatmapList-class](#).

### Value

A [unit](#) object

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
# no example for this internal method
```

---

component\_width-dispatch

*Method dispatch page for component\_width*

---

### Description

Method dispatch page for component\_width.

### Dispatch

component\_width can be dispatched on following classes:

- [component\\_width,HeatmapList-method](#), [HeatmapList-class](#) class method
- [component\\_width,Heatmap-method](#), [Heatmap-class](#) class method

### Examples

```
# no example  
NULL
```

---

component\_width-Heatmap-method

*Width of each heatmap component*

---

### Description

Width of each heatmap component

### Usage

```
## S4 method for signature 'Heatmap'  
component_width(object, k = 1:7)
```

### Arguments

object            a [Heatmap-class](#) object.  
k                 which component in the heatmap, see [Heatmap-class](#).

### Details

This function is only for internal use.

### Value

A [unit](#) object.

### Details

This function is only for internal use.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
# no example for this internal method
```

---

component\_width-HeatmapList-method

*Width of each heatmap list component*

---

### Description

Width of each heatmap list component

### Usage

```
## S4 method for signature 'HeatmapList'  
component_width(object, k = 1:7)
```

### Arguments

object            a [HeatmapList-class](#) object.  
k                 which component in the heatmap list, see [HeatmapList-class](#).

### Details

This function is only for internal use.

### Value

A [unit](#) object

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
# no example for this internal method
```

---

decorate\_annotation    *Decorate the heatmap annotation*

---

### Description

Decorate the heatmap annotation

### Usage

```
decorate_annotation(annotation, code, slice, envir = new.env(parent = parent.frame()))
```

**Arguments**

annotation	name of the annotation
code	code that adds graphics in the selected heatmap body
slice	index of row slices in the heatmap
envir	where to look for variables inside code

**Details**

There is a viewport for every column annotation and row annotation. This function constructs the name of the viewport, goes to the viewport by `seekViewport` and applies code to that viewport.

**Value**

The function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
set.seed(123)
ha1 = HeatmapAnnotation(df = data.frame(type = rep(letters[1:2], 5)))
ha2 = rowAnnotation(point = anno_points(runif(10), which = "row"))
Heatmap(matrix(rnorm(100), 10), name = "mat", km = 2,
  top_annotation = ha1) + ha2
decorate_annotation("type", {
  grid.circle(x = unit(c(0.2, 0.4, 0.6, 0.8), "npc"),
    gp = gpar(fill = "#FF000080"))
})
decorate_annotation("point", {
  grid.rect(gp = gpar(fill = "#FF000080"))
}, slice = 2)
```

---

decorate\_column\_dend *Decorate heatmap dendrogram on columns*

---

**Description**

Decorate heatmap dendrogram on columns

**Usage**

```
decorate_column_dend(..., envir = new.env(parent = parent.frame()))
```

**Arguments**

...	pass to <code>decorate_dend</code>
envir	where to look for variables inside code

**Details**

This is a wrapper function which pre-defined which argument in [decorate\\_dend](#).

**Value**

The function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# No example for this function
NULL
```

---

decorate\_column\_names *Decorate heatmap column names*

---

**Description**

Decorate heatmap column names

**Usage**

```
decorate_column_names(..., envir = new.env(parent = parent.frame()))
```

**Arguments**

...	pass to <a href="#">decorate_dimnames</a>
envir	where to look for variables inside code

**Details**

This is a helper function which pre-defined which argument in [decorate\\_dimnames](#).

**Value**

The function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# No example for this function
NULL
```

---

decorate\_column\_title *Decorate heatmap column title*

---

### Description

Decorate heatmap column title

### Usage

```
decorate_column_title(..., envir = new.env(parent = parent.frame()))
```

### Arguments

... pass to [decorate\\_title](#)  
envir where to look for variables inside code

### Details

This is a helper function which pre-defined which argument in [decorate\\_title](#).

### Value

The function returns no value.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
# No example for this function  
NULL
```

---

decorate\_dend *Decorate the heatmap dendrogram*

---

### Description

Decorate the heatmap dendrogram

### Usage

```
decorate_dend(heatmap, code, slice = 1, which = c("column", "row"),  
  envir = new.env(parent = parent.frame()))
```

**Arguments**

heatmap	name of the heatmap
code	code that adds graphics in the selected heatmap body
slice	index of row slices in the heatmap
which	on rows or on columns?
envir	where to look for variables inside code

**Details**

There is a viewport for each dendrogram in the heatmap. This function constructs the name of the viewport, goes to the viewport by `seekViewport` and applies code to that viewport.

If you know the number of leaves in the dendrogram, it is simple to calculate the position of every leaf in the dendrogram. E.g., for the column dendrogram, the  $i^{\text{th}}$  leaf is located at:

```
# assume nc is the number of columns
unit((i-0.5)/nc, "npc")
```

**Value**

This function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
set.seed(123)
Heatmap(matrix(rnorm(100), 10), name = "mat", km = 2)
decorate_dend("mat", {
  grid.rect(gp = gpar(fill = "#FF000080"))
}, which = "row", slice = 2)
```

---

decorate_dimnames	<i>Decorate the heatmap dimension names</i>
-------------------	---

---

**Description**

Decorate the heatmap dimension names

**Usage**

```
decorate_dimnames(heatmap, code, slice = 1, which = c("column", "row"),
  envir = new.env(parent = parent.frame()))
```

**Arguments**

heatmap	name of the heatmap
code	code that adds graphics in the selected heatmap body
slice	index of row slices in the heatmap
which	on rows or on columns?
envir	where to look for variables inside code

## Details

There is a viewport for row names and column names in the heatmap. This function constructs the name of the viewport, goes to the viewport by `seekViewport` and applies code to that viewport.

If you know the dimensions of the matrix, it is simple to calculate the position of every row name or column name in the heatmap. E.g., for the column column, the  $i^{\text{th}}$  name is located at:

```
# assume nc is the number of columns
unit((i-0.5)/nc, "npc")
```

## Value

The function returns no value.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
set.seed(123)
mat = matrix(rnorm(100), 10)
rownames(mat) = letters[1:10]
colnames(mat) = LETTERS[1:10]
Heatmap(mat, name = "mat", km = 2)

decorate_dimnames("mat", {
  grid.rect(gp = gpar(fill = "#FF000080"))
}, which = "row", slice = 2)
```

---

decorate\_heatmap\_body *Decorate the heatmap body*

---

## Description

Decorate the heatmap body

## Usage

```
decorate_heatmap_body(heatmap, code, slice = 1, envir = new.env(parent = parent.frame()))
```

## Arguments

heatmap	name of the heatmap which is set as name option in <code>Heatmap</code> function
code	code that adds graphics in the selected heatmap body
slice	index of row slices in the heatmap if it is split by rows
envir	where to look for variables inside code



**Details**

There is a viewport for each row slice in each heatmap. This function constructs the name of the viewport, goes to the viewport by [seekViewport](#) and applies code to that viewport.

If you know the number of rows and columns for that row slice, it is simple to calculate the position of every small grid in the row slice. E.g., the position for the grid in  $i^{\text{th}}$  row and  $j^{\text{th}}$  column is:

```
# assume nc is the number of columns
# and nr is the number of rows in that row slice
unit((i-0.5)/nc, "npc")
unit((j-0.5)/nr, "npc")

# the width is
unit(1/nc, "npc")

# the height is
unit(1/nr, "npc")
```

**Value**

This function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
set.seed(123)
Heatmap(matrix(rnorm(100), 10), name = "mat")
decorate_heatmap_body("mat", {
  grid.circle(gp = gpar(fill = "#FF000080"))
})
```

---

decorate_row_dend	<i>Decorate heatmap dendrogram on rows</i>
-------------------	--

---

**Description**

Decorate heatmap dendrogram on rows

**Usage**

```
decorate_row_dend(..., envir = new.env(parent = parent.frame()))
```

**Arguments**

...	pass to <a href="#">decorate_dend</a>
envir	where to look for variables inside code

**Details**

This is a helper function which pre-defined which argument in [decorate\\_dend](#).

**Value**

The function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# No example for this function
NULL
```

---

decorate_row_names	<i>Decorate heatmap row names</i>
--------------------	-----------------------------------

---

**Description**

Decorate heatmap row names

**Usage**

```
decorate_row_names(..., envir = new.env(parent = parent.frame()))
```

**Arguments**

...	pass to <a href="#">decorate_dimnames</a>
envir	where to look for variables inside code

**Details**

This is a helper function which pre-defined which argument in [decorate\\_dimnames](#).

**Value**

The function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# No example for this function
NULL
```

---

decorate_row_title	<i>Decorate heatmap row title</i>
--------------------	-----------------------------------

---

**Description**

Decorate heatmap row title

**Usage**

```
decorate_row_title(..., envir = new.env(parent = parent.frame()))
```

**Arguments**

...	pass to <a href="#">decorate_title</a>
envir	where to look for variables inside code

**Details**

This is a helper function which pre-defined which argument in [decorate\\_title](#).

**Value**

The function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# No example for this function  
NULL
```

---

decorate_title	<i>Decorate the heatmap title</i>
----------------	-----------------------------------

---

**Description**

Decorate the heatmap title

**Usage**

```
decorate_title(heatmap, code, slice = 1, which = c("column", "row"),  
  envir = new.env(parent = parent.frame()))
```

**Arguments**

heatmap	name of the heatmap
code	code that adds graphics in the selected heatmap body
slice	index of row slices in the heatmap
which	on rows or on columns?
envir	where to look for variables inside code

**Details**

There is a viewport for row titles and column title in the heatmap. This function constructs the name of the viewport, goes to the viewport by `seekViewport` and applies code to that viewport.

**Value**

The function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
set.seed(123)
Heatmap(matrix(rnorm(100), 10), name = "mat", km = 2)
decorate_title("mat", {
  grid.rect(gp = gpar(fill = "#FF000080"))
}, which = "row", slice = 2)
```

---

densityHeatmap

*Use colors to represent density distribution*

---

**Description**

Use colors to represent density distribution

**Usage**

```
densityHeatmap(data,
  col = rev(brewer.pal(11, "Spectral")),
  density_param = list(na.rm = TRUE),
  color_space = "LAB",
  anno = NULL,
  ylab = deparse(substitute(data)),
  title = paste0("Density heatmap of ", deparse(substitute(data))),
  range = c(-Inf, Inf),
  cluster_columns = FALSE,
  clustering_distance_columns = "euclidean",
  clustering_method_columns = "complete",
  column_dend_side = "top",
  column_dend_height = unit(10, "mm"),
```

```

show_column_dend = FALSE,
column_dend_gp = gpar(),
column_dend_reorder = TRUE,
column_names_side = c("bottom", "top"),
show_column_names = TRUE,
column_names_max_height = unit(4, "cm"),
column_names_gp = gpar(fontsize = 12),
column_order = NULL,
...)
```

### Arguments

<code>data</code>	a matrix or a list. If it is a matrix, density will be calculated by columns.
<code>col</code>	a list of colors that density values are mapped to.
<code>density_param</code>	parameters send to <a href="#">density</a> , <code>na.rm</code> is enforced to TRUE.
<code>color_space</code>	the color space in which colors are interpolated. Pass to <a href="#">colorRamp2</a> .
<code>anno</code>	annotation for the matrix columns or the list. The value should be a vector or a data frame and colors for annotations are randomly assigned. If you want to customize the annotation colors, use a <a href="#">HeatmapAnnotation-class</a> object directly.
<code>ylab</code>	label on y-axis in the plot
<code>title</code>	title of the plot
<code>range</code>	ranges on the y-axis. By default the range is between 1th quantile and 99th quantile of the data.
<code>cluster_columns</code>	whether cluster columns (here cluster by density distributions)
<code>clustering_distance_columns</code>	pass to <a href="#">Heatmap</a>
<code>clustering_method_columns</code>	pass to <a href="#">Heatmap</a>
<code>column_dend_side</code>	pass to <a href="#">Heatmap</a>
<code>column_dend_height</code>	pass to <a href="#">Heatmap</a>
<code>show_column_dend</code>	pass to <a href="#">Heatmap</a>
<code>column_dend_gp</code>	pass to <a href="#">Heatmap</a>
<code>column_dend_reorder</code>	pass to <a href="#">Heatmap</a>
<code>column_names_side</code>	pass to <a href="#">Heatmap</a>
<code>show_column_names</code>	pass to <a href="#">Heatmap</a>
<code>column_names_max_height</code>	pass to <a href="#">Heatmap</a>
<code>column_names_gp</code>	pass to <a href="#">Heatmap</a>
<code>column_order</code>	order of columns
<code>...</code>	pass to <a href="#">draw</a> , <a href="#">HeatmapList-method</a>

**Details**

To visualize data distribution in a matrix or in a list, sometimes we use boxplot or beanplot. Here we use colors to map the density values and visualize distribution of values in each column (or each vector in the list) through a heatmap. It is useful if you have huge number of columns in data to visualize.

The density matrix is generated with 500 rows ranging between the maximum and minimal values in all densities. The density values in each row are linearly interpolated between the two density values at the two nearest bounds.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
matrix = matrix(rnorm(100), 10); colnames(matrix) = letters[1:10]
densityHeatmap(matrix)
densityHeatmap(matrix, anno = rep(c("A", "B"), each = 5))
densityHeatmap(matrix, col = c("white", "red"), anno = rep(c("A", "B"), each = 5))

ha = HeatmapAnnotation(points = anno_points(runif(10)),
  anno = rep(c("A", "B"), each = 5), col = list(anno = c("A" = "red", "B" = "blue")))
densityHeatmap(matrix, anno = ha)

lt = list(rnorm(10), rnorm(10))
densityHeatmap(lt)
```

---

dist2

---

*Calculate pairwise distance from a matrix*


---

**Description**

Calculate pairwise distance from a matrix

**Usage**

```
dist2(mat, pairwise_fun = function(x, y) sqrt(sum((x - y)^2)), ...)
```

**Arguments**

mat	a matrix. The distance is calculated by rows.
pairwise_fun	a function which calculates distance between two vectors.
...	pass to <a href="#">as.dist</a> .

**Details**

You can construct any type of distance measurements by defining a pair-wise distance function. The function is implemented by two nested for loops, so the efficiency may not be so good.

**Value**

A `dist` object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
mat = matrix(rnorm(40), nr = 4, ncol = 10)
rownames(mat) = letters[1:4]
colnames(mat) = letters[1:10]

d2 = dist2(mat)
d2 = dist2(mat, pairwise_fun = function(x, y) 1 - cor(x, y))
# distance only calculated within 10 and 90 quantile of each vector
d2 = dist2(mat, pairwise_fun = function(x, y) {
  q1 = quantile(x, c(0.1, 0.9))
  q2 = quantile(y, c(0.1, 0.9))
  l = x > q1[1] & x < q1[2] & y > q2[1] & y < q2[2]
  sqrt(sum((x[l] - y[l])^2))
})
```

---

draw-dispatch

*Method dispatch page for draw*

---

**Description**

Method dispatch page for draw.

**Dispatch**

draw can be dispatched on following classes:

- [draw, HeatmapAnnotation-method, HeatmapAnnotation-class](#) class method
- [draw, SingleAnnotation-method, SingleAnnotation-class](#) class method
- [draw, HeatmapList-method, HeatmapList-class](#) class method
- [draw, Heatmap-method, Heatmap-class](#) class method

**Examples**

```
# no example
NULL
```

---

draw-Heatmap-method    *Draw a single heatmap*

---

## Description

Draw a single heatmap

## Usage

```
## S4 method for signature 'Heatmap'  
draw(object, internal = FALSE, test = FALSE, ...)
```

## Arguments

object	a <a href="#">Heatmap-class</a> object.
internal	only used inside the calling of <a href="#">draw,HeatmapList-method</a> . Only heatmap without legends will be drawn.
test	only for testing
...	pass to <a href="#">draw,HeatmapList-method</a> .

## Details

The function creates a [HeatmapList-class](#) object which only contains a single heatmap and call [draw,HeatmapList-method](#) to make the final heatmap.

## Value

This function returns no value.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
mat = matrix(rnorm(80, 2), 8, 10)  
mat = rbind(mat, matrix(rnorm(40, -2), 4, 10))  
rownames(mat) = letters[1:12]  
colnames(mat) = letters[1:10]  
  
ht = Heatmap(mat)  
draw(ht, heatmap_legend_side = "left")
```



---

`draw-HeatmapAnnotation-method`*Draw the heatmap annotations*

---

## Description

Draw the heatmap annotations

## Usage

```
## S4 method for signature 'HeatmapAnnotation'  
draw(object, index, k = NULL, n = NULL, align_to = "bottom", ...)
```

## Arguments

<code>object</code>	a <a href="#">HeatmapAnnotation-class</a> object.
<code>index</code>	a vector of order.
<code>k</code>	if row annotation is splitted, the value identifies which row slice.
<code>n</code>	total number of row slices.
<code>align_to</code>	if the allocated space is more than than the column annotation itself, should the viewport be aligned to the top or bottom?
<code>...</code>	pass to <a href="#">viewport</a> which contains all annotations.

## Details

A viewport is created. Mostly, this method is used inside [draw,HeatmapList-method](#).

## Value

No value is returned.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
df = data.frame(type = c("a", "a", "a", "b", "b", "b"))  
ha = HeatmapAnnotation(df = df)  
grid.newpage(); draw(ha, 1:6)  
grid.newpage(); draw(ha, 6:1)  
  
ha = HeatmapAnnotation(df = df, col = list(type = c("a" = "red", "b" = "blue")))  
grid.newpage(); draw(ha, 1:6)  
  
ha = HeatmapAnnotation(df = df, col = list(type = c("a" = "red", "b" = "blue")),  
  which = "row")  
grid.newpage(); draw(ha, 1:6)  
  
ha = HeatmapAnnotation(points = anno_points(1:6))  
grid.newpage(); draw(ha, 1:6)
```

```

ha = HeatmapAnnotation(histogram = anno_barplot(1:6))
grid.newpage(); draw(ha, 1:6)

mat = matrix(rnorm(36), 6)
ha = HeatmapAnnotation(boxplot = anno_boxplot(mat))
grid.newpage(); draw(ha, 1:6)

```

---

draw-HeatmapList-method

*Draw a list of heatmaps*

---

## Description

Draw a list of heatmaps

## Usage

```

## S4 method for signature 'HeatmapList'
draw(object,
      padding = unit(c(2, 2, 2, 2), "mm"),
      newpage = TRUE,
      row_title = character(0),
      row_title_side = c("left", "right"),
      row_title_gp = gpar(fontsize = 14),
      column_title = character(0),
      column_title_side = c("top", "bottom"),
      column_title_gp = gpar(fontsize = 14),
      heatmap_legend_side = c("right", "left", "bottom", "top"),
      show_heatmap_legend = TRUE,
      heatmap_legend_list = list(),
      annotation_legend_side = c("right", "left", "bottom", "top"),
      show_annotation_legend = TRUE,
      annotation_legend_list = list(),
      gap = unit(3, "mm"),
      main_heatmap = which(sapply(object@ht_list, inherits, "Heatmap"))[1],
      row_dend_side = c("original", "left", "right"),
      row_sub_title_side = c("original", "left", "right"), ...)

```

## Arguments

object	a <a href="#">HeatmapList-class</a> object
padding	padding of the plot. Elements correspond to bottom, left, top, right paddings.
newpage	whether create a new page for the graphics.
row_title	title on the row.
row_title_side	will the title be put on the left or right of the heatmap.
row_title_gp	graphic parameters for drawing text.
column_title	title on the column.

`column_title_side` will the title be put on the top or bottom of the heatmap.  
`column_title_gp` graphic parameters for drawing text.  
`heatmap_legend_side` side of the heatmap legend.  
`show_heatmap_legend` whether show heatmap legend.  
`heatmap_legend_list` a list of self-defined legend, should be wrapped into `grob` objects.  
`annotation_legend_side` side of annotation legend.  
`show_annotation_legend` whether show annotation legend.  
`annotation_legend_list` a list of self-defined legend, should be wrapped into `grob` objects.  
`gap` gap between heatmaps, should be a `unit` object.  
`main_heatmap` name or index for the main heatmap  
`row_dend_side` if auto adjust, where to put the row dendrograms for the main heatmap  
`row_sub_title_side` if auto adjust, where to put sub row titles for the main heatmap  
`...` pass to `make_layout, HeatmapList-method`

## Details

The function first calls `make_layout, HeatmapList-method` to calculate the layout of the heatmap list and the layout of every single heatmap, then makes the plot by re-calling the graphic functions which are already recorded in the layout.

## Value

This function returns a list of row dendrograms and column dendrogram.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```

mat = matrix(rnorm(80, 2), 8, 10)
mat = rbind(mat, matrix(rnorm(40, -2), 4, 10))
rownames(mat) = letters[1:12]
colnames(mat) = letters[1:10]

ht = Heatmap(mat)
ht_list = ht + ht
draw(ht_list)
draw(ht_list, row_title = "row title", column_title = "column title",
heatmap_legend_side = "top")

```

---

 draw-SingleAnnotation-method

*Draw the single annotation*


---

## Description

Draw the single annotation

## Usage

```
## S4 method for signature 'SingleAnnotation'
draw(object, index, k = NULL, n = NULL)
```

## Arguments

object	a <a href="#">SingleAnnotation-class</a> object.
index	a vector of orders
k	if row annotation is splitted, the value identifies which row slice. It is only used for the names of the viewport which contains the annotation graphics.
n	total number of row slices

## Details

A viewport is created.

The graphics would be different depending the annotation is a row annotation or a column annotation.

## Value

No value is returned.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
anno = SingleAnnotation(name = "test", value = c("a", "a", "a", "b", "b", "b"))
grid.newpage(); draw(anno, 1:5)
grid.newpage(); draw(anno, c(1, 4, 3, 5, 2))
```

```
anno = SingleAnnotation(value = c("a", "a", "a", "b", "b", "b"),
  col = c("a" = "red", "b" = "blue"))
grid.newpage(); draw(anno, 1:5)
grid.newpage(); draw(anno, c(1, 4, 3, 5, 2))
```

```
anno = SingleAnnotation(value = c("a", "a", "a", "b", "b", "b"),
  col = c("a" = "red", "b" = "blue"), which = "row")
grid.newpage(); draw(anno, 1:5)
```

```
anno = SingleAnnotation(value = 1:10)
grid.newpage(); draw(anno, 1:10)
```

```
require(circlize)
anno = SingleAnnotation(value = 1:10, col = colorRamp2(c(1, 10), c("blue", "red")))
grid.newpage(); draw(anno, 1:10)

anno = SingleAnnotation(fun = anno_points(1:10))
grid.newpage(); draw(anno, 1:10)
```

---

draw\_annotation-Heatmap-method  
*Draw column annotations*

---

## Description

Draw column annotations

## Usage

```
## S4 method for signature 'Heatmap'
draw_annotation(object, which = c("top", "bottom"))
```

## Arguments

**object** a [Heatmap-class](#) object.  
**which** are the annotations put on the top or bottom of the heatmap?

## Details

A viewport is created which contains column annotations.

Since the column annotations is a [HeatmapAnnotation-class](#) object, the function calls [draw, HeatmapAnnotation-met](#) to draw the annotations.

This function is only for internal use.

## Value

This function returns no value.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
# no example for this internal method
NULL
```

---

draw\_annotation\_legend-HeatmapList-method  
*Draw legends for all column annotations*

---

### Description

Draw legends for all column annotations

### Usage

```
## S4 method for signature 'HeatmapList'
draw_annotation_legend(object, legend_list = list(), ...)
```

### Arguments

object	a <a href="#">HeatmapList-class</a> object
legend_list	a list of self-defined legend, should be wrapped into <a href="#">grob</a> objects.
...	graphic parameters passed to <a href="#">color_mapping_legend, ColorMapping-method</a> .

### Details

A viewport is created which contains annotation legends.  
 This function is only for internal use.

### Value

This function returns no value.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
# no example for this internal method
NULL
```

---

draw\_dend-Heatmap-method  
*Draw dendrogram on row or column*

---

### Description

Draw dendrogram on row or column

### Usage

```
## S4 method for signature 'Heatmap'
draw_dend(object,
  which = c("row", "column"), k = 1, max_height = NULL, ...)
```

**Arguments**

object	a <a href="#">Heatmap-class</a> object.
which	is dendrogram put on the row or on the column of the heatmap?
k	a matrix may be splitted by rows, the value identifies which row-slice.
max_height	maximum height of the dendrograms.
...	pass to <a href="#">viewport</a> , basically for defining the position of the viewport.

**Details**

If the matrix is split into several row slices, a list of dendrograms will be drawn by the heatmap that each dendrogram corresponds to its row slices.

A viewport is created which contains dendrograms.

This function is only for internal use.

**Value**

This function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

[grid.dendrogram](#)

**Examples**

```
# There is no example
NULL
```

---

draw\_dimnames-Heatmap-method

*Draw row names or column names*

---

**Description**

Draw row names or column names

**Usage**

```
## S4 method for signature 'Heatmap'
draw_dimnames(object,
  which = c("row", "column"), k = 1, dimname_padding = unit(0, "mm"), ...)
```

**Arguments**

object            a [Heatmap-class](#) object.  
which            are names put on the row or on the column of the heatmap?  
k                a matrix may be split by rows, the value identifies which row-slice.  
dimname\_padding   padding for the row/column names  
...              pass to [viewport](#), basically for defining the position of the viewport.

**Details**

A viewport is created which contains row names or column names.  
This function is only for internal use.

**Value**

This function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this internal method  
NULL
```

---

draw\_heatmap\_body-Heatmap-method  
*Draw the heatmap body*

---

**Description**

Draw the heatmap body

**Usage**

```
## S4 method for signature 'Heatmap'  
draw_heatmap_body(object, k = 1, ...)
```

**Arguments**

object            a [Heatmap-class](#) object.  
k                a matrix may be split by rows, the value identifies which row-slice.  
...              pass to [viewport](#), basically for defining the position of the viewport.



**Details**

The matrix can be split into several parts by rows if `km` or `split` is specified when initializing the `Heatmap` object. If the matrix is split, there will be gaps between rows to identify different row-slice.

A viewport is created which contains subset rows of the heatmap.

This function is only for internal use.

**Value**

This function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this internal method
NULL
```

---

draw\_heatmap\_legend-HeatmapList-method

*Draw legends for all heatmaps*

---

**Description**

Draw legends for all heatmaps

**Usage**

```
## S4 method for signature 'HeatmapList'
draw_heatmap_legend(object, legend_list = list(), ...)
```

**Arguments**

<code>object</code>	a <code>HeatmapList-class</code> object
<code>legend_list</code>	a list of self-defined legend, should be wrapped into <code>grob</code> objects.
<code>...</code>	graphic parameters passed to <code>color_mapping_legend, ColorMapping-method</code> .

**Details**

A viewport is created which contains heatmap legends.

This function is only for internal use.

**Value**

This function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this internal method
NULL
```

---

draw\_heatmap\_list-HeatmapList-method  
*Draw the list of heatmaps*

---

**Description**

Draw the list of heatmaps

**Usage**

```
## S4 method for signature 'HeatmapList'
draw_heatmap_list(object)
```

**Arguments**

object            a [HeatmapList-class](#) object

**Details**

A viewport is created which contains heatmaps.

This function is only for internal use.

**Value**

This function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this internal method
NULL
```

---

draw\_title-dispatch    *Method dispatch page for draw\_title*

---

### Description

Method dispatch page for draw\_title.

### Dispatch

draw\_title can be dispatched on following classes:

- [draw\\_title, HeatmapList-method, HeatmapList-class](#) class method
- [draw\\_title, Heatmap-method, Heatmap-class](#) class method

### Examples

```
# no example
NULL
```

---

draw\_title-Heatmap-method  
*Draw heatmap title*

---

### Description

Draw heatmap title

### Usage

```
## S4 method for signature 'Heatmap'
draw_title(object,
  which = c("row", "column"), k = 1, ...)
```

### Arguments

object	a <a href="#">Heatmap-class</a> object.
which	is title put on the row or on the column of the heatmap?
k	a matrix may be split by rows, the value identifies which row-slice.
...	pass to <a href="#">viewport</a> , basically for defining the position of the viewport.

### Details

A viewport is created which contains heatmap title.  
This function is only for internal use.

### Value

This function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this internal method
NULL
```

---

draw\_title-HeatmapList-method

*Draw heatmap list title*

---

**Description**

Draw heatmap list title

**Usage**

```
## S4 method for signature 'HeatmapList'
draw_title(object,
  which = c("column", "row"))
```

**Arguments**

object            a [HeatmapList-class](#) object  
which            dendrogram on the row or on the column of the heatmap

**Details**

A viewport is created which contains heatmap list title.

This function is only for internal use.

**Value**

This function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this internal method
NULL
```

---

enhanced\_basicplot      *Enhanced version of basic barplot and boxplot*

---

### Description

Enhanced version of basic barplot and boxplot

### Usage

```
enhanced_basicplot(data, ..., ylim = NULL,
  ylab = deparse(substitute(data)), title = NULL, title_gp = gpar(fontsize = 14),
  type = c("boxplot", "barplot"), width = 0.8, gp = gpar(),
  pch = 1, size = unit(2, "mm"), axis_gp = gpar(fontsize = 8),
  padding = unit(c(2, 18, 2, 2), "mm"),
  heatmap_legend_list = list())
```

### Arguments

data	a matrix, a list or a simple numeric vector. If your data is a data frame please convert it to a matrix in the first place.
...	pass to <a href="#">Heatmap</a>
ylim	ranges on y axis
ylab	label on y axis
title	title of the plot
title_gp	graphic parameters for the title
type	type of the plot
width	relative width of the bar or box
gp	graphic parameters for hte bar or box
pch	shape of outlier points in the boxplot
size	size of hte outlier points in the boxplot
axis_gp	graphic parameters for the axis
padding	padding of the plot
heatmap_legend_list	a list of <a href="#">grob</a> which contains legend. It can be generated by <a href="#">color_mapping_legend</a> , <a href="#">ColorMapping</a>

### Details

This function adds annotations to the barplot or boxplot.

This function is still quite experimental.

### Value

No value is returned

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
mat = matrix(runif(100), 10)
enhanced_basicplot(mat)
ha = HeatmapAnnotation(char = sample(letters[1:2], 10, replace = TRUE),
                       num = runif(10))
enhanced_basicplot(mat, top_annotation = ha)
enhanced_basicplot(mat, type = "barplot", top_annotation = ha)
```

---

get\_color\_mapping\_list-HeatmapAnnotation-method  
*Get a list of color mapping objects*

---

## Description

Get a list of color mapping objects

## Usage

```
## S4 method for signature 'HeatmapAnnotation'
get_color_mapping_list(object)
```

## Arguments

object            a [HeatmapAnnotation-class](#) object.

## Details

Color mapping for visible simple annotations are only returned.

This function is only for internal use.

## Value

A list of [ColorMapping-class](#) objects or an empty list.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
# no example for this internal method
NULL
```

---

get\_color\_mapping\_param\_list-HeatmapAnnotation-method  
*Get a list of color mapping parameters*

---

**Description**

Get a list of color mapping parameters

**Usage**

```
## S4 method for signature 'HeatmapAnnotation'  
get_color_mapping_param_list(object)
```

**Arguments**

object            a [HeatmapAnnotation-class](#) object.

**Details**

Color mapping parameters for visible simple annotations are only returned.

This function is only for internal use.

**Value**

A list.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example  
NULL
```

---

grid.dendrogram            *Draw dendrogram under grid system*

---

**Description**

Draw dendrogram under grid system

**Usage**

```
grid.dendrogram(dend, facing = c("bottom", "top", "left", "right"),  
                 max_height = NULL, order = c("normal", "reverse"), ...)
```

**Arguments**

dend	a <a href="#">dendrogram</a> object.
facing	facing of the dendrogram.
max_height	maximum height of the dendrogram. It is useful to make dendrograms comparable if you want to plot more than one dendrograms. Height for each dendrogram can be obtained by <code>attr(dend, "height")</code> .
order	should leaves of dendrogram be put in the normal order (1, ..., n) or reverse order (n, ..., 1)? It may matters for the dendrograms putting on left and right.
...	pass to <a href="#">viewport</a> which contains the dendrogram.

**Details**

The dendrogram can be rendered (e.g. by `dendextend` package).

A viewport is created which contains the dendrogram.

This function only plots the dendrogram without adding labels. The leaves of the dendrogram locates at `unit(c(0.5, 1.5, ... (n-0.5))/n, "npc")`.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
hc = hclust(dist(USArrests[1:5, ]))
dend = as.dendrogram(hc)

grid.newpage()
layout = grid.layout(nrow = 2, ncol = 2)
pushViewport(viewport(layout = layout))
grid.dendrogram(dend, layout.pos.row = 1, layout.pos.col = 1)
grid.dendrogram(dend, facing = "top", layout.pos.row = 1, layout.pos.col = 2)
grid.dendrogram(dend, facing = "top", order = "reverse", layout.pos.row = 2,
  layout.pos.col = 1)
grid.dendrogram(dend, facing = "left", layout.pos.row = 2, layout.pos.col = 2)
upViewport()
```

---

grid.dendrogram2

*Draw dendrogram under grid system*

---

**Description**

Draw dendrogram under grid system

**Usage**

```
grid.dendrogram2(dend, facing = c("bottom", "top", "left", "right"),
  max_height = NULL, order = c("normal", "reverse"), ...)
```



**Arguments**

dend	a <code>dendrogram</code> object which has been adjusted by <code>adjust_dend_by_leaf_width</code> , or else it will be sent back to <code>grid.dendrogram</code> .
facing	same as in <code>grid.dendrogram</code> .
max_height	same as in <code>grid.dendrogram</code> .
order	same as in <code>grid.dendrogram</code> .
...	same as in <code>grid.dendrogram</code> .

**Author(s)**

Zuguang gu <z.gu@dkfz.de>

**Examples**

```
m = matrix(rnorm(100), 10)
dend = as.dendrogram(hclust(dist(m)))
dend = adjust_dend_by_leaf_width(dend, width = 1:10)
grid.dendrogram2(dend)
```

---

Heatmap

*Constructor method for Heatmap class*


---

**Description**

Constructor method for Heatmap class

**Usage**

```
Heatmap(matrix, col, name,
  na_col = "grey",
  color_space = "LAB",
  rect_gp = gpar(col = NA),
  cell_fun = NULL,
  row_title = character(0),
  row_title_side = c("left", "right"),
  row_title_gp = gpar(fontsize = 14),
  row_title_rot = switch(row_title_side[1], "left" = 90, "right" = 270),
  column_title = character(0),
  column_title_side = c("top", "bottom"),
  column_title_gp = gpar(fontsize = 14),
  column_title_rot = 0,
  cluster_rows = TRUE,
  clustering_distance_rows = "euclidean",
  clustering_method_rows = "complete",
  row_dend_side = c("left", "right"),
  row_dend_width = unit(10, "mm"),
  show_row_dend = TRUE,
  row_dend_reorder = TRUE,
  row_dend_gp = gpar(),
  row_hclust_side = row_dend_side,
```

```

row_hclust_width = row_dend_width,
show_row_hclust = show_row_dend,
row_hclust_reorder = row_dend_reorder,
row_hclust_gp = row_dend_gp,
cluster_columns = TRUE,
clustering_distance_columns = "euclidean",
clustering_method_columns = "complete",
column_dend_side = c("top", "bottom"),
column_dend_height = unit(10, "mm"),
show_column_dend = TRUE,
column_dend_gp = gpar(),
column_dend_reorder = TRUE,
column_hclust_side = column_dend_side,
column_hclust_height = column_dend_height,
show_column_hclust = show_column_dend,
column_hclust_gp = column_dend_gp,
column_hclust_reorder = column_dend_reorder,
row_order = NULL,
column_order = NULL,
row_names_side = c("right", "left"),
show_row_names = TRUE,
row_names_max_width = default_row_names_max_width(),
row_names_gp = gpar(fontsize = 12),
column_names_side = c("bottom", "top"),
show_column_names = TRUE,
column_names_max_height = default_column_names_max_height(),
column_names_gp = gpar(fontsize = 12),
top_annotation = new("HeatmapAnnotation"),
top_annotation_height = top_annotation@size,
bottom_annotation = new("HeatmapAnnotation"),
bottom_annotation_height = bottom_annotation@size,
km = 1,
km_title = "cluster%i",
split = NULL,
gap = unit(1, "mm"),
combined_name_fun = function(x) paste(x, collapse = "/"),
width = NULL,
show_heatmap_legend = TRUE,
heatmap_legend_param = list(title = name),
use_raster = FALSE,
raster_device = c("png", "jpeg", "tiff", "CairoPNG", "CairoJPEG", "CairoTIFF"),
raster_quality = 2,
raster_device_param = list()

```

### Arguments

- |                     |   |
|---------------------|---|
| <code>matrix</code> | a matrix. Either numeric or character. If it is a simple vector, it will be converted to a one-column matrix.   |
| <code>col</code>    | a vector of colors if the color mapping is discrete or a color mapping function if the matrix is continuous numbers (should be generated by <a href="#">colorRamp2</a> ). If the matrix is continuous, the value can also be a vector of colors so that colors will be interpolated. Pass to <a href="#">ColorMapping</a> . |

<code>name</code>	name of the heatmap. The name is used as the title of the heatmap legend.
<code>na_col</code>	color for NA values.
<code>rect_gp</code>	graphic parameters for drawing rectangles (for heatmap body).
<code>color_space</code>	the color space in which colors are interpolated. Only used if <code>matrix</code> is numeric and <code>col</code> is a vector of colors. Pass to <a href="#">colorRamp2</a> .
<code>cell_fun</code>	self-defined function to add graphics on each cell. Seven parameters will be passed into this function: <code>i</code> , <code>j</code> , <code>x</code> , <code>y</code> , <code>width</code> , <code>height</code> , <code>fill</code> which are row index, column index in <code>matrix</code> , coordinate of the middle points in the heatmap body viewport, the width and height of the cell and the filled color. <code>x</code> , <code>y</code> , <code>width</code> and <code>height</code> are all <a href="#">unit</a> objects.
<code>row_title</code>	title on row.
<code>row_title_side</code>	will the title be put on the left or right of the heatmap?
<code>row_title_gp</code>	graphic parameters for drawing text.
<code>row_title_rot</code>	rotation of row titles. Only 0, 90, 270 are allowed to set.
<code>column_title</code>	title on column.
<code>column_title_side</code>	will the title be put on the top or bottom of the heatmap?
<code>column_title_gp</code>	graphic parameters for drawing text.
<code>column_title_rot</code>	rotation of column titles. Only 0, 90, 270 are allowed to set.
<code>cluster_rows</code>	If the value is a logical, it means whether make cluster on rows. The value can also be a <a href="#">hclust</a> or a <a href="#">dendrogram</a> that already contains clustering information. This means you can use any type of clustering methods and render the <a href="#">dendrogram</a> object with self-defined graphic settings.
<code>clustering_distance_rows</code>	it can be a pre-defined character which is in ("euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski", "pearson", "spearman", "kendall"). It can also be a function. If the function has one argument, the input argument should be a matrix and the returned value should be a <a href="#">dist</a> object. If the function has two arguments, the input arguments are two vectors and the function calculates distance between these two vectors.
<code>clustering_method_rows</code>	method to make cluster, pass to <a href="#">hclust</a> .
<code>row_dend_side</code>	should the row cluster be put on the left or right of the heatmap?
<code>row_dend_width</code>	width of the row cluster, should be a <a href="#">unit</a> object.
<code>show_row_dend</code>	whether show row clusters.
<code>row_dend_gp</code>	graphics parameters for drawing lines. If users already provide a <a href="#">dendrogram</a> object with edges rendered, this argument will be ignored.
<code>row_dend_reorder</code>	apply reordering on rows. The value can be a logical value or a vector which contains weight which is used to reorder rows
<code>row_hclust_side</code>	deprecated, use <code>row_dend_side</code> instead
<code>row_hclust_width</code>	deprecated, use <code>row_dend_width</code> instead

<code>show_row_hclust</code>	deprecated, use <code>show_row_dend</code> instead
<code>row_hclust_gp</code>	deprecated, use <code>row_dend_gp</code> instead
<code>row_hclust_reorder</code>	deprecated, use <code>row_dend_reorder</code> instead
<code>cluster_columns</code>	whether make cluster on columns. Same settings as <code>cluster_rows</code> .
<code>clustering_distance_columns</code>	same setting as <code>clustering_distance_rows</code> .
<code>clustering_method_columns</code>	method to make cluster, pass to <code>hclust</code> .
<code>column_dend_side</code>	should the column cluster be put on the top or bottom of the heatmap?
<code>column_dend_height</code>	height of the column cluster, should be a <code>unit</code> object.
<code>show_column_dend</code>	whether show column clusters.
<code>column_dend_gp</code>	graphic parameters for drawing lines. Same settings as <code>row_dend_gp</code> .
<code>column_dend_reorder</code>	apply reordering on columns. The value can be a logical value or a vector which contains weight which is used to reorder columns
<code>column_hclust_side</code>	deprecated, use <code>column_dend_side</code> instead
<code>column_hclust_height</code>	deprecated, use <code>column_dend_height</code> instead
<code>show_column_hclust</code>	deprecated, use <code>show_column_dend</code> instead
<code>column_hclust_gp</code>	deprecated, use <code>column_dend_gp</code> instead
<code>column_hclust_reorder</code>	deprecated, use <code>column_dend_reorder</code> instead
<code>row_order</code>	order of rows. It makes it easy to adjust row order for a list of heatmaps if this heatmap is selected as the main heatmap. Manually setting row order should turn off clustering
<code>column_order</code>	order of column. It makes it easy to adjust column order for both matrix and column annotations.
<code>row_names_side</code>	should the row names be put on the left or right of the heatmap?
<code>show_row_names</code>	whether show row names.
<code>row_names_max_width</code>	maximum width of row names viewport. Because some times row names can be very long, it is not reasonable to show them all.
<code>row_names_gp</code>	graphic parameters for drawing text.
<code>column_names_side</code>	should the column names be put on the top or bottom of the heatmap?
<code>column_names_max_height</code>	maximum height of column names viewport.
<code>show_column_names</code>	whether show column names.

<code>column_names_gp</code>	graphic parameters for drawing text.
<code>top_annotation</code>	a <a href="#">HeatmapAnnotation</a> object which contains a list of annotations.
<code>top_annotation_height</code>	total height of the column annotations on the top.
<code>bottom_annotation</code>	a <a href="#">HeatmapAnnotation</a> object.
<code>bottom_annotation_height</code>	total height of the column annotations on the bottom.
<code>km</code>	do k-means clustering on rows. If the value is larger than 1, the heatmap will be split by rows according to the k-means clustering. For each row-clusters, hierarchical clustering is still applied with parameters above.
<code>km_title</code>	row title for each cluster when km is set. It must a text with format of ".*%i.*" where "%i" is replaced by the index of the cluster.
<code>split</code>	a vector or a data frame by which the rows are split. But if <code>cluster_rows</code> is a clustering object, <code>split</code> can be a single number indicating rows are to be split according to the split on the tree.
<code>gap</code>	gap between row-slices if the heatmap is split by rows, should be <a href="#">unit</a> object. If it is a vector, the order corresponds to top to bottom in the heatmap
<code>combined_name_fun</code>	if the heatmap is split by rows, how to make a combined row title for each slice? The input parameter for this function is a vector which contains level names under each column in <code>split</code> .
<code>width</code>	the width of the single heatmap, should be a fixed <a href="#">unit</a> object. It is used for the layout when the heatmap is appended to a list of heatmaps.
<code>show_heatmap_legend</code>	whether show heatmap legend?
<code>heatmap_legend_param</code>	a list contains parameters for the heatmap legend. See <a href="#">color_mapping_legend</a> , <a href="#">ColorMapping-method</a> for all available parameters.
<code>use_raster</code>	whether render the heatmap body as a raster image. It helps to reduce file size when the matrix is huge. Note if <code>cell_fun</code> is set, <code>use_raster</code> is enforced to be FALSE.
<code>raster_device</code>	graphic device which is used to generate the raster image
<code>raster_quality</code>	a value set to larger than 1 will improve the quality of the raster image.
<code>raster_device_param</code>	a list of further parameters for the selected graphic device

## Details

The initialization function only applies parameter checking and fill values to each slot with proper ones. Then it will be ready for clustering and layout.

Following methods can be applied on the [Heatmap-class](#) object:

- [show, Heatmap-method](#): draw a single heatmap with default parameters
- [draw, Heatmap-method](#): draw a single heatmap.
- [add\\_heatmap, Heatmap-method](#) append heatmaps and row annotations to a list of heatmaps.

The constructor function pretends to be a high-level graphic function because the `show` method of the [Heatmap-class](#) object actually plots the graphics.

**Value**

A `Heatmap-class` object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
mat = matrix(rnorm(80, 2), 8, 10)
mat = rbind(mat, matrix(rnorm(40, -2), 4, 10))
rownames(mat) = letters[1:12]
colnames(mat) = letters[1:10]

require(circlize)

Heatmap(mat)
Heatmap(mat, col = colorRamp2(c(-3, 0, 3), c("green", "white", "red")))
Heatmap(mat, name = "test")
Heatmap(mat, column_title = "blablabla")
Heatmap(mat, row_title = "blablabla")
Heatmap(mat, column_title = "blablabla", column_title_side = "bottom")
Heatmap(mat, column_title = "blablabla", column_title_gp = gpar(fontsize = 20,
  fontface = "bold"))
Heatmap(mat, cluster_rows = FALSE)
Heatmap(mat, clustering_distance_rows = "pearson")
Heatmap(mat, clustering_distance_rows = function(x) dist(x))
Heatmap(mat, clustering_distance_rows = function(x, y) 1 - cor(x, y))
Heatmap(mat, clustering_method_rows = "single")
Heatmap(mat, row_dend_side = "right")
Heatmap(mat, row_dend_width = unit(1, "cm"))
Heatmap(mat, row_names_side = "left", row_dend_side = "right",
  column_names_side = "top", column_dend_side = "bottom")
Heatmap(mat, show_row_names = FALSE)

mat2 = mat
rownames(mat2) = NULL
colnames(mat2) = NULL
Heatmap(mat2)

Heatmap(mat, row_names_gp = gpar(fontsize = 20))
Heatmap(mat, km = 2)
Heatmap(mat, split = rep(c("A", "B"), 6))
Heatmap(mat, split = data.frame(rep(c("A", "B"), 6), rep(c("C", "D"), each = 6)))
Heatmap(mat, split = data.frame(rep(c("A", "B"), 6), rep(c("C", "D"), each = 6)),
  combined_name_fun = function(x) paste(x, collapse = "\n"))

annotation = HeatmapAnnotation(df = data.frame(type = c(rep("A", 6), rep("B", 6))))
Heatmap(mat, top_annotation = annotation)

annotation = HeatmapAnnotation(df = data.frame(type1 = rep(c("A", "B"), 6),
  type2 = rep(c("C", "D"), each = 6)))
Heatmap(mat, bottom_annotation = annotation)

annotation = data.frame(value = rnorm(10))
annotation = HeatmapAnnotation(df = annotation)
```

```
Heatmap(mat, top_annotation = annotation)

annotation = data.frame(value = rnorm(10))
value = 1:10
ha = HeatmapAnnotation(df = annotation, points = anno_points(value),
  annotation_height = c(1, 2))
Heatmap(mat, top_annotation = ha, top_annotation_height = unit(2, "cm"),
  bottom_annotation = ha)

# character matrix
mat3 = matrix(sample(letters[1:6], 100, replace = TRUE), 10, 10)
rownames(mat3) = {x = letters[1:10]; x[1] = "aaaaaaaaaaaaaaaaaaaaa";x}
Heatmap(mat3, rect_gp = gpar(col = "white"))

mat = matrix(1:9, 3, 3)
rownames(mat) = letters[1:3]
colnames(mat) = letters[1:3]

Heatmap(mat, rect_gp = gpar(col = "white"),
  cell_fun = function(i, j, x, y, width, height, fill) {
    grid.text(mat[i, j], x = x, y = y)
  },
  cluster_rows = FALSE, cluster_columns = FALSE, row_names_side = "left",
  column_names_side = "top")
```

Heatmap-class

*Class for a single heatmap***Description**

Class for a single heatmap

**Details**

The components for a single heatmap are placed into a 9 x 7 layout:

```

+-----+ (1)
+-----+ (2)
+-----+ (3)
+-----+ (4)
+---+---+-----+---+---+
|1|2|3| 4(5) |5|6|7|
+---+---+-----+---+---+
+-----+ (6)
+-----+ (7)
+-----+ (8)
+-----+ (9)
```

From top to bottom in column 4, the regions are:

- title which is put on the top of the heatmap, graphics are drawn by [draw\\_title](#), [Heatmap-method](#).
- column cluster on the top, graphics are drawn by [draw\\_dend](#), [Heatmap-method](#).

- column annotation on the top, graphics are drawn by [draw\\_annotation,Heatmap-method](#).
- column names on the top, graphics are drawn by [draw\\_dimnames,Heatmap-method](#).
- heatmap body, graphics are drawn by [draw\\_heatmap\\_body,Heatmap-method](#).
- column names on the bottom, graphics are drawn by [draw\\_dimnames,Heatmap-method](#).
- column annotation on the bottom, graphics are drawn by [draw\\_annotation,Heatmap-method](#).
- column cluster on the bottom, graphics are drawn by [draw\\_dend,Heatmap-method](#).
- title on the bottom, graphics are drawn by [draw\\_title,Heatmap-method](#).

From left to right in row 5, the regions are:

- title which is put in the left of the heatmap, graphics are drawn by [draw\\_title,Heatmap-method](#).
- row cluster on the left, graphics are drawn by [draw\\_dend,Heatmap-method](#).
- row names on the left, graphics are drawn by [draw\\_dimnames,Heatmap-method](#).
- heatmap body
- row names on the right, graphics are drawn by [draw\\_dimnames,Heatmap-method](#).
- row cluster on the right, graphics are drawn by [draw\\_dend,Heatmap-method](#).
- title on the right, graphics are drawn by [draw\\_title,Heatmap-method](#).

The [Heatmap-class](#) is not responsible for heatmap legend and annotation legends. The [draw,Heatmap-method](#) method will construct a [HeatmapList-class](#) object which only contains one single heatmap and call [draw,HeatmapList-method](#) to make a complete heatmap.

## Methods

The [Heatmap-class](#) provides following methods:

- [Heatmap](#): constructor method.
- [draw,Heatmap-method](#): draw a single heatmap.
- [add\\_heatmap,Heatmap-method](#) append heatmaps and row annotations to a list of heatmaps.
- [row\\_order,HeatmapList-method](#): get order of rows
- [column\\_order,HeatmapList-method](#): get order of columns
- [row\\_dend,HeatmapList-method](#): get row dendrograms
- [column\\_dend,HeatmapList-method](#): get column dendrograms

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
# for examples, please go to `Heatmap` method page
NULL
```



---

HeatmapAnnotation      *Constructor method for HeatmapAnnotation class*

---

## Description

Constructor method for HeatmapAnnotation class

## Usage

```
HeatmapAnnotation(df, name, col, na_col = "grey",
  annotation_legend_param = list(),
  show_legend = TRUE,
  ...,
  which = c("column", "row"),
  annotation_height = 1,
  annotation_width = 1,
  height = calc_anno_size(),
  width = calc_anno_size(),
  gp = gpar(col = NA),
  gap = unit(0, "mm"),
  show_annotation_name = FALSE,
  annotation_name_gp = gpar(),
  annotation_name_offset = unit(2, "mm"),
  annotation_name_side = ifelse(which == "column", "right", "bottom"),
  annotation_name_rot = ifelse(which == "column", 0, 90))
```

## Arguments

df	a data frame. Each column will be treated as a simple annotation. The data frame must have column names.
name	name of the heatmap annotation, optional.
col	a list of colors which contains color mapping to columns in df. See <a href="#">SingleAnnotation</a> for how to set colors.
na_col	color for NA values in simple annotations.
annotation_legend_param	a list which contains parameters for annotation legends
show_legend	whether show legend for each column in df.
...	functions which define complex annotations or vectors of simple annotation. Values should be named arguments.
which	are the annotations row annotations or column annotations?
annotation_height	height of each annotation if annotations are column annotations.
annotation_width	width of each annotation if annotations are row annotations.
height	height of the column annotations, basically it is identical to <code>bottom_annotation_height</code> or <code>top_annotation_height</code> in <a href="#">Heatmap</a> function.
width	width of the whole heatmap annotations, only used for row annotation when appending to the list of heatmaps.

<code>gp</code>	graphic parameters for simple annotations.
<code>gap</code>	gap between each annotation
<code>show_annotation_name</code>	whether show annotation names. For column annotation, annotation names are drawn either on the left or the right, and for row annotations, names are drawn either on top to at bottom. The value can be a vector.
<code>annotation_name_gp</code>	graphic parameters for annotation names. Graphic parameters can be vectors.
<code>annotation_name_offset</code>	offset to the annotations, <code>unit</code> object. The value can be a vector.
<code>annotation_name_side</code>	side of the annotation names.
<code>annotation_name_rot</code>	rotation of the annotation names, can only take values in <code>c(00, 90, 180, 270)</code> . The value can be a vector.

### Details

The simple annotations are defined by `df` and `col` arguments. Complex annotations are defined by the function list. So you need to at least to define `df` or a annotation function.

### Value

A `HeatmapAnnotation-class` object.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### See Also

There are two shortcut functions: `rowAnnotation` and `columnAnnotation`.

### Examples

```
df = data.frame(type = c("a", "a", "a", "b", "b", "b"))
ha = HeatmapAnnotation(df = df)

ha = HeatmapAnnotation(df = df, col = list(type = c("a" = "red", "b" = "blue")))
ha = HeatmapAnnotation(type = c("a", "a", "a", "b", "b", "b"),
  col = list(type = c("a" = "red", "b" = "blue")))

ha = HeatmapAnnotation(df = df, col = list(type = c("a" = "red", "b" = "blue")),
  which = "row")

ha = HeatmapAnnotation(points = anno_points(1:6))

ha = HeatmapAnnotation(histogram = anno_points(1:6))

mat = matrix(rnorm(36), 6)
ha = HeatmapAnnotation(boxplot = anno_boxplot(mat))
```

---

HeatmapAnnotation-class

*Class for heatmap annotations*

---

### Description

Class for heatmap annotations

### Details

A complex heatmap contains a list of annotations which are represented as different graphics placed on rows and columns. The [HeatmapAnnotation-class](#) contains a list of single annotations which are represented as a list of [SingleAnnotation-class](#) objects with same number of rows or columns.

### Methods

The [HeatmapAnnotation-class](#) provides following methods:

- [HeatmapAnnotation](#): constructor method
- [draw,HeatmapAnnotation-method](#): draw the annotations

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
# for examples, please go to `HeatmapAnnotation` method page  
NULL
```

---

HeatmapList

*Constructor method for HeatmapList class*

---

### Description

Constructor method for HeatmapList class

### Usage

```
HeatmapList(...)
```

### Arguments

```
...           arguments
```

### Details

There is no public constructor method for the [HeatmapList-class](#).

**Value**

No value is returned.

**Details**

There is no public constructor method for the [HeatmapList-class](#).

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example
NULL
```

---

HeatmapList-class      *Class for a list of heatmaps*

---

**Description**

Class for a list of heatmaps

**Details**

A heatmap list is defined as a list of heatmaps and row annotations.

The components for the heatmap list are placed into a 7 x 7 layout:

```

+-----+(1)
+-----+(2)
+-----+(3)
+---+-----+---+
|1|2|3| 4(4) |5|6|7|
+---+-----+---+
+-----+(5)
+-----+(6)
+-----+(7)

```

From top to bottom in column 4, the regions are:

- annotation legend on the top, graphics are drawn by [draw\\_annotation\\_legend,HeatmapList-method](#).
- heatmap legend on the top, graphics are drawn by [draw\\_heatmap\\_legend,HeatmapList-method](#).
- title for the heatmap list which is put on the top, graphics are drawn by [draw\\_title,HeatmapList-method](#).
- the list of heatmaps and row annotations
- title for the heatmap list which is put on the bottom, graphics are drawn by [draw\\_title,HeatmapList-method](#).
- heatmap legend on the bottom, graphics are drawn by [draw\\_heatmap\\_legend,HeatmapList-method](#).
- annotation legend on the bottom, graphics are drawn by [draw\\_annotation\\_legend,HeatmapList-method](#).

From left to right in row 4, the regions are:

- annotation legend on the left, graphics are drawn by `draw_annotation_legend,HeatmapList-method`.
- heatmap legend on the left, graphics are drawn by `draw_heatmap_legend,HeatmapList-method`.
- title for the heatmap list which is put on the left, graphics are drawn by `draw_title,HeatmapList-method`.
- the list of heatmaps and row annotations
- title for the heatmap list which is put on the right, graphics are drawn by `draw_title,HeatmapList-method`.
- heatmap legend on the right, graphics are drawn by `draw_heatmap_legend,HeatmapList-method`.
- annotation legend on the right, graphics are drawn by `draw_annotation_legend,HeatmapList-method`.

For the list of heatmaps which are placed at (5, 5) in the layout, the heatmaps and row annotations are placed one after the other.

## Methods

The `HeatmapList-class` provides following methods:

- `draw,HeatmapList-method`: draw the list of heatmaps and row annotations.
- `add_heatmap,HeatmapList-method`: add heatmaps to the list of heatmaps.
- `row_order,HeatmapList-method`: get order of rows
- `column_order,HeatmapList-method`: get order of columns
- `row_dend,HeatmapList-method`: get row dendrograms
- `column_dend,HeatmapList-method`: get column dendrograms

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
mat = matrix(rnorm(80, 2), 8, 10)
mat = rbind(mat, matrix(rnorm(40, -2), 4, 10))
rownames(mat) = letters[1:12]
colnames(mat) = letters[1:10]

ht = Heatmap(mat)
ht + ht
ht + ht + ht

ht_list = ht + ht
ht + ht_list

ha = HeatmapAnnotation(points = anno_points(1:12, which = "row"),
  which = "row")
ht + ha
ht_list + ha
```

---

heatmap\_legend\_size-HeatmapList-method  
*Size of the heatmap legend viewport*

---

**Description**

Size of the heatmap legend viewport

**Usage**

```
## S4 method for signature 'HeatmapList'
heatmap_legend_size(object, legend_list = list(), ...)
```

**Arguments**

object            a [HeatmapList-class](#) object  
 legend\_list      a list of self-defined legend, should be wrapped into [grob](#) objects.  
 ...                graphic parameters passed to [color\\_mapping\\_legend,ColorMapping-method](#).

**Details**

This function is only for internal use.

**Value**

A [unit](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this internal method
NULL
```

---

ht\_global\_opt            *Global graphic options for heatmaps*

---

**Description**

Global graphic options for heatmaps

**Usage**

```
ht_global_opt(..., RESET = FALSE, READ.ONLY = NULL, LOCAL = FALSE, ADD = FALSE)
```

**Arguments**

...	options, see 'details' section
RESET	reset all the option values
READ.ONLY	TRUE means only to return read-only values, FALSE means only to return non-read-only values, NULL means to return both.
LOCAL	switch local mode
ADD	add new options

**Details**

You can set some parameters for all heatmaps/annotations simultaneously by this global function. Pleast note you should put it before your heatmap code and reset all option values after drawing the heatmaps to get rid of affecting next heatmap plotting.

There are following parameters:

**heatmap\_row\_names\_gp** set row\_names\_gp in [Heatmap](#).  
**heatmap\_column\_names\_gp** set column\_names\_gp in [Heatmap](#).  
**heatmap\_row\_title\_gp** set row\_title\_gp in [Heatmap](#).  
**heatmap\_column\_title\_gp** set column\_title\_gp in [Heatmap](#).  
**heatmap\_legend\_title\_gp** set title\_gp element in heatmap\_legend\_param in [Heatmap](#).  
**heatmap\_legend\_title\_position** set title\_position element in heatmap\_legend\_param in [Heatmap](#).  
**heatmap\_legend\_labels\_gp** set labels\_gp element in heatmap\_legend\_param in [Heatmap](#).  
**heatmap\_legend\_grid\_width** set grid\_width element in heatmap\_legend\_param in [Heatmap](#).  
**heatmap\_legend\_grid\_height** set grid\_height element in heatmap\_legend\_param in [Heatmap](#).  
**heatmap\_legend\_grid\_border** set grid\_border element in heatmap\_legend\_param in [Heatmap](#).  
**heatmap\_legend\_title\_gp** set title\_gp element in legend\_param in [SingleAnnotation](#).  
**heatmap\_legend\_title\_position** set title\_position element in legend\_param in [SingleAnnotation](#).  
**heatmap\_legend\_labels\_gp** set labels\_gp element in legend\_param in [SingleAnnotation](#).  
**heatmap\_legend\_grid\_width** set grid\_width element in legend\_param in [SingleAnnotation](#).  
**heatmap\_legend\_grid\_height** set grid\_height element in legend\_param in [SingleAnnotation](#).  
**heatmap\_legend\_grid\_border** set grid\_border element in legend\_param in [SingleAnnotation](#).  
**fast\_hclust** whether use [hclust](#) to speed up clustering?

You can get or set option values by the traditional way (like [options](#)) or by \$ operator:

```
# to get option values
ht_global_opt("heatmap_row_names_gp")
ht_global_opt$heatmap_row_names_gp

# to set option values
ht_global_opt("heatmap_row_names_gp" = gpar(fontsize = 8))
ht_global_opt$heatmap_row_names_gp = gpar(fontsize = 8)
```

**Value**

Depends on the options users selected.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this function
NULL
```

---

is\_abs\_unit

*Whether the unit object contains absolute unit*

---

**Description**

Whether the unit object contains absolute unit

**Usage**

```
is_abs_unit(u)
```

**Arguments**

u                    a [unit](#) object

**Details**

Besides the normal absolute units (e.g. "mm", "inches"), this function simply treat [grob](#) objects as absolute units.

For a complex unit which is combination of different units, it is absolute only if all units included are absolute units.

**Value**

A logical value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
is_abs_unit(unit(1, "mm"))
is_abs_unit(unit(1, "npc"))
is_abs_unit(textGrob("foo"))
is_abs_unit(unit(1, "mm") + unit(1, "npc"))
```



Legend

*Making legend grobs***Description**

Making legend grobs

**Usage**

```
Legend(at, labels = at, nrow = NULL, ncol = 1, col_fun, by_row = FALSE,
       grid_height = unit(4, "mm"), grid_width = unit(4, "mm"), gap = unit(2, "mm"),
       labels_gp = gpar(fontsize = 10),
       border = NULL, background = "#EEEEEE",
       type = "grid", legend_gp = gpar(),
       pch = 16, size = unit(2, "mm"),
       legend_height = NULL, legend_width = NULL,
       direction = c("vertical", "horizontal"),
       title = "", title_gp = gpar(fontsize = 10, fontface = "bold"),
       title_position = c("topleft", "topcenter", "leftcenter", "lefttop"))
```

**Arguments**

at	breaks, can be wither numeric or character
labels	labels corresponding to at
nrow	if there are too many legends, they can be positioned in an array, this controls number of rows
ncol	if there are too many legends, they can be positioned in an array, this controls number of columns. At a same time only one of nrow and ncol can be specified.
col_fun	a color mapping function which is used to make a continuous color bar
by_row	when there are multiple columns for legends, whether to arrange them by rows.
grid_height	height of legend grid
grid_width	width of legend grid
gap	when legends are put in multiple columns, this is the gap between neighbouring columns, measured as a <code>unit</code> object
labels_gp	graphic parameters for labels
border	color of legend borders, also for the ticks in the continuous legend
background	background colors
type	type of legends, can be grid, points and lines
legend_gp	graphic parameters for the legend
pch	type of points
size	size of points
legend_height	height of the whole legend, used when col_fun is specified and direction is set to vertical
legend_width	width of the whole legend, used when col_fun is specified and direction is set to horizontal

direction direction of the continuous or discrete legend  
 title title of the legend  
 title\_gp graphic parameters of title  
 title\_position position of title according to the legend

**Value**

A [grob](#) object

**See Also**

[packLegend](#) packs multiple legends into one [grob](#) object

**Examples**

```
lgd = Legend(title = "discrete", at = 1:4, labels = letters[1:4],
  legend_gp = gpar(fill = 2:5))
grid.newpage()
grid.draw(lgd)

require(circlize)
col_fun = colorRamp2(c(-1, 0, 1), c("blue", "white", "red"))
lgd = Legend(title = "continuous", at = seq(-1, 1, by = 0.5), col_fun = col_fun)
grid.newpage()
grid.draw(lgd)

lgd = Legend(title = "continuous", at = seq(-1, 1, by = 0.5), col_fun = col_fun,
  direction = "horizontal")
grid.newpage()
grid.draw(lgd)

lgd = Legend(title = "discrete", at = 1:10, labels = letters[1:10],
  ncol = 4, by_row = TRUE, legend_gp = gpar(fill = rand_color(10)))
grid.newpage()
grid.draw(lgd)

lgd = Legend(title = "lty", at = 1:3, labels = 1:3, type = "lines",
  legend_gp = gpar(lty = 1:3))
grid.newpage()
grid.draw(lgd)
```

---

make\_column\_cluster-Heatmap-method

*Make cluster on columns*

---

**Description**

Make cluster on columns

**Usage**

```
## S4 method for signature 'Heatmap'
make_column_cluster(object)
```

**Arguments**

object            a [Heatmap-class](#) object.

**Details**

The function will fill or adjust column\_dend and column\_order slots.

This function is only for internal use.

**Value**

A [Heatmap-class](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this internal method
NULL
```

---

make\_layout-dispatch    *Method dispatch page for make\_layout*

---

**Description**

Method dispatch page for make\_layout.

**Dispatch**

make\_layout can be dispatched on following classes:

- [make\\_layout, HeatmapList-method, HeatmapList-class](#) class method
- [make\\_layout, Heatmap-method, Heatmap-class](#) class method

**Examples**

```
# no example
NULL
```

---

make\_layout-Heatmap-method

*Make the layout of a single heatmap*

---

### Description

Make the layout of a single heatmap

### Usage

```
## S4 method for signature 'Heatmap'  
make_layout(object)
```

### Arguments

object            a [Heatmap-class](#) object.

### Details

The layout of the single heatmap will be established by setting the size of each heatmap components. Also functions that make graphics for heatmap components will be recorded.

Whether apply row clustering or column clustering affects the layout, so clustering should be applied first before making the layout.

This function is only for internal use.

### Value

A [Heatmap-class](#) object.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
# no example for this internal method  
NULL
```

---

make\_layout-HeatmapList-method

*Make layout for the complete plot*

---

### Description

Make layout for the complete plot

**Usage**

```
## S4 method for signature 'HeatmapList'
make_layout(object, row_title = character(0),
  row_title_side = c("left", "right"),
  row_title_gp = gpar(fontsize = 14),
  column_title = character(0),
  column_title_side = c("top", "bottom"),
  column_title_gp = gpar(fontsize = 14),
  heatmap_legend_side = c("right", "left", "bottom", "top"),
  merge_legends = FALSE,
  show_heatmap_legend = TRUE,
  heatmap_legend_list = list(),
  annotation_legend_side = c("right", "left", "bottom", "top"),
  show_annotation_legend = TRUE,
  annotation_legend_list = list(),
  gap = unit(3, "mm"),
  row_gap = NULL,
  main_heatmap = which(sapply(object@ht_list, inherits, "Heatmap"))[1],
  row_dend_side = c("original", "left", "right"),
  row_hclust_side = row_dend_side,
  row_sub_title_side = c("original", "left", "right"),
  cluster_rows = NULL,
  clustering_distance_rows = NULL,
  clustering_method_rows = NULL,
  row_dend_width = NULL,
  show_row_dend = NULL,
  row_dend_reorder = NULL,
  row_dend_gp = NULL,
  row_order = NULL,
  km = NULL,
  split = NULL,
  combined_name_fun = NULL)
```

**Arguments**

**object** a [HeatmapList-class](#) object.

**row\_title** title on the row.

**row\_title\_side** will the title be put on the left or right of the heatmap.

**row\_title\_gp** graphic parameters for drawing text.

**column\_title** title on the column.

**column\_title\_side** will the title be put on the top or bottom of the heatmap.

**column\_title\_gp** graphic parameters for drawing text.

**heatmap\_legend\_side** side of the heatmap legend.

**merge\_legends** whether put heatmap legends and annotation legends in a same column

**show\_heatmap\_legend** whether show heatmap legend.

heatmap_legend_list	a list of self-defined legend, should be wrapped into <a href="#">grob</a> objects.
annotation_legend_side	side of annotation legend.
show_annotation_legend	whether show annotation legend.
annotation_legend_list	a list of self-defined legend, should be wrapped into <a href="#">grob</a> objects.
gap	gap between heatmaps, should be a <a href="#">unit</a> object.
row_gap	gap between row clusters if rows are split
main_heatmap	name or index for the main heatmap
row_dend_side	if auto adjust, where to put the row dendrograms for the main heatmap
row_hclust_side	deprecated, use row_dend_side instead
row_sub_title_side	if auto adjust, where to put sub row titles for the main heatmap
cluster_rows	same setting as in <a href="#">Heatmap</a> , if it is specified, cluster_rows in main heatmap is ignored.
clustering_distance_rows	same setting as in <a href="#">Heatmap</a> , if it is specified, clustering_distance_rows in main heatmap is ignored.
clustering_method_rows	same setting as in <a href="#">Heatmap</a> , if it is specified, clustering_method_rows in main heatmap is ignored.
row_dend_width	same setting as in <a href="#">Heatmap</a> , if it is specified, row_dend_width in main heatmap is ignored.
show_row_dend	same setting as in <a href="#">Heatmap</a> , if it is specified, show_row_dend in main heatmap is ignored.
row_dend_reorder	same setting as in <a href="#">Heatmap</a> , if it is specified, row_dend_reorder in main heatmap is ignored.
row_dend_gp	same setting as in <a href="#">Heatmap</a> , if it is specified, row_dend_gp in main heatmap is ignored.
row_order	same setting as in <a href="#">Heatmap</a> , if it is specified, row_order in main heatmap is ignored.
km	same setting as in <a href="#">Heatmap</a> , if it is specified, km in main heatmap is ignored.
split	same setting as in <a href="#">Heatmap</a> , if it is specified, split in main heatmap is ignored.
combined_name_fun	same setting as in <a href="#">Heatmap</a> , if it is specified, combined_name_fun in main heatmap is ignored.

## Details

It sets the size of each component of the heatmap list and adjusts graphic parameters for each heatmap if necessary.

The layout for the heatmap list and layout for each heatmap are calculated when drawing the heatmap list.

This function is only for internal use.

**Value**

A [HeatmapList-class](#) object in which settings for each heatmap are adjusted.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this internal method
NULL
```

---

make\_row\_cluster-Heatmap-method  
*Make cluster on rows*

---

**Description**

Make cluster on rows

**Usage**

```
## S4 method for signature 'Heatmap'
make_row_cluster(object)
```

**Arguments**

object            a [Heatmap-class](#) object.

**Details**

The function will fill or adjust row\_dend\_list, row\_order\_list, row\_title and matrix\_param slots.

If order is defined, no clustering will be applied.

This function is only for internal use.

**Value**

A [Heatmap-class](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this internal method
NULL
```

---

map\_to\_colors-ColorMapping-method  
*Map values to colors*

---

## Description

Map values to colors

## Usage

```
## S4 method for signature 'ColorMapping'  
map_to_colors(object, x)
```

## Arguments

object	a <a href="#">ColorMapping-class</a> object.
x	input values.

## Details

It maps a vector of values to a vector of colors.

## Value

A vector of colors.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## Examples

```
# discrete color mapping for characters  
cm = ColorMapping(name = "test",  
  colors = c("blue", "white", "red"),  
  levels = c("a", "b", "c"))  
map_to_colors(cm, "a")  
map_to_colors(cm, c("a", "a", "b"))  
  
# discrete color mapping for numeric values  
cm = ColorMapping(name = "test",  
  colors = c("blue", "white", "red"),  
  levels = c(1, 2, 3))  
map_to_colors(cm, 1)  
map_to_colors(cm, "1")  
map_to_colors(cm, c(1, 1, 2, 2))  
  
# continuous color mapping  
require(circlize)  
cm = ColorMapping(name = "test",  
  col_fun = colorRamp2(c(0, 0.5, 1), c("blue", "white", "red")))  
map_to_colors(cm, 0.2)  
map_to_colors(cm, seq(0.2, 0.8, by = 0.1))
```



---

max_text_height	<i>Maximum height of text</i>
-----------------	-------------------------------

---

**Description**

Maximum height of text

**Usage**

```
max_text_height(text, ...)
```

**Arguments**

text	a vector of text
...	pass to <a href="#">textGrob</a>

**Details**

Simply calculate maximum height of a list of [textGrob](#) objects.

**Value**

A [unit](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

[max\\_text\\_height](#) is always used to calculate the size of viewport when there is text annotation ([anno\\_text](#))

**Examples**

```
x = c("a", "b\nb", "c\nc\nc")
max_text_height(x, gp = gpar(fontsize = 10))
```

---

max_text_width	<i>Maximum width of text</i>
----------------	------------------------------

---

**Description**

Maximum width of text

**Usage**

```
max_text_width(text, ...)
```

**Arguments**

text            a vector of text  
 ...            pass to `textGrob`

**Details**

Simply calculate maximum width of a list of `textGrob` objects.

**Value**

A `unit` object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**See Also**

`max_text_width` is always used to calculate the size of viewport when there is text annotation (`anno_text`)

**Examples**

```
x = c("a", "bb", "ccc")
max_text_width(x, gp = gpar(fontsize = 10))
```

---

oncoPrint

*Make oncoPrint*

---

**Description**

Make oncoPrint

**Usage**

```
oncoPrint(mat, get_type = function(x) x,
  alter_fun = alter_fun_list, alter_fun_list = NULL, col,
  row_order = oncoprint_row_order(),
  column_order = oncoprint_column_order(),
  show_pct = TRUE, pct_gp = row_names_gp, pct_digits = 0,
  axis_gp = gpar(fontsize = 8),
  show_row_barplot = TRUE,
  row_barplot_width = unit(2, "cm"),
  remove_empty_columns = FALSE,
  heatmap_legend_param = list(title = "Alterations"),
  top_annotation = HeatmapAnnotation(column_bar = anno_oncoprint_barplot(),
  annotation_height = unit(2, "cm")),
  top_annotation_height = top_annotation@size,
  bottom_annotation = new("HeatmapAnnotation"),
  bottom_annotation_height = bottom_annotation@size,
  barplot_ignore = NULL,
```

```

row_title = character(),
row_title_side = c("left", "right"),
row_title_gp = gpar(fontsize = 14),
row_title_rot = switch(row_title_side[1], "left" = 90, "right" = 270),
column_title = character(),
column_title_side = c("top", "bottom"),
column_title_gp = gpar(fontsize = 14),
column_title_rot = 0,
show_row_names = TRUE,
row_names_gp = gpar(fontsize = 12),
show_column_names = FALSE,
column_names_gp = gpar(fontsize = 12),
split = NULL,
gap = unit(1, "mm"),
combined_name_fun = function(x) paste(x, collapse = "/"),
width = NULL,
...)

```

### Arguments

mat	a character matrix which encodes multiple alterations or a list of matrix for which every matrix contains binary value representing the alteration is present or absent. When it is a list, the names represent alteration types. You can use <a href="#">unify_mat_list</a> to make all matrix having same row names and column names.
get_type	If different alterations are encoded in the matrix, this self-defined function determines how to extract them. Only work when mat is a matrix.
alter_fun	a single function or a list of functions which define how to add graphics for different alterations. If it is a list, the names of the list should cover all alteration types.
alter_fun_list	deprecated, use alter_run instead.
col	a vector of color for which names correspond to alteration types.
row_order	order of genes. By default it is sorted by frequency of alterations decreasingly. Set it to NULL if you don't want to set the order
column_order	order of samples. By default the order is calculated by the 'memo sort' method which can visualize the mutual exclusivity across genes. Set it to NULL if you don't want to set the order
show_pct	whether show percent values on the left of the oncoprint
pct_gp	graphic parameters for percent row annotation
pct_digits	digits for percent values
axis_gp	graphic parameters for axes
show_row_barplot	whether show barplot annotation on rows
row_barplot_width	width of barplot annotation on rows. It should be a <a href="#">unit</a> object
remove_empty_columns	if there is no alteration in that sample, whether remove it on the heatmap
heatmap_legend_param	pass to <a href="#">Heatmap</a>

<code>top_annotation</code>	by default the top annotation contains barplots representing frequency of mutations in every sample.
<code>top_annotation_height</code>	total height of the column annotations on the top.
<code>bottom_annotation</code>	a <a href="#">HeatmapAnnotation</a> object.
<code>bottom_annotation_height</code>	total height of the column annotations on the bottom.
<code>barplot_ignore</code>	specific alterations that you don't want to put on the barplots. If you want to really suppress the top barplot set <code>top_annotation</code> to NULL.
<code>row_title</code>	title on row.
<code>row_title_side</code>	will the title be put on the left or right of the heatmap?
<code>row_title_gp</code>	graphic parameters for drawing text.
<code>row_title_rot</code>	rotation of row titles. Only 0, 90, 270 are allowed to set.
<code>column_title</code>	title on column.
<code>column_title_side</code>	will the title be put on the top or bottom of the heatmap?
<code>column_title_gp</code>	graphic parameters for drawing text.
<code>column_title_rot</code>	rotation of column titles. Only 0, 90, 270 are allowed to set.
<code>show_row_names</code>	whether show row names.
<code>row_names_gp</code>	graphic parameters for drawing text.
<code>show_column_names</code>	whether show column names.
<code>column_names_gp</code>	graphic parameters for drawing text.
<code>split</code>	a vector or a data frame by which the rows are split. But if <code>cluster_rows</code> is a clustering object, <code>split</code> can be a single number indicating rows are to be split according to the split on the tree.
<code>gap</code>	gap between row-slices if the heatmap is split by rows, should be <a href="#">unit</a> object. If it is a vector, the order corresponds to top to bottom in the heatmap
<code>combined_name_fun</code>	if the heatmap is split by rows, how to make a combined row title for each slice? The input parameter for this function is a vector which contains level names under each column in <code>split</code> .
<code>width</code>	the width of the single heatmap, should be a fixed <a href="#">unit</a> object. It is used for the layout when the heatmap is appended to a list of heatmaps.
<code>...</code>	pass to <a href="#">Heatmap</a> , so can set <code>bottom_annotation</code> here.

## Details

The function returns a normal heatmap list and you can add more heatmaps/row annotations to it.

The 'memo sort' method is from <https://gist.github.com/armish/564a65ab874a770e2c26>. Thanks to B. Arman Aksoy for contributing the code.

The function would be a little bit slow if you plot it in an interactive device because all alterations are added through a `for` loop.

For more explanation, please go to the vignette.

**Value**

A `HeatmapList-class` object which means you can add other heatmaps or row annotations to it.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example
NULL
```

---

packLegend	<i>Pack legends</i>
------------	---------------------

---

**Description**

Pack legends

**Usage**

```
packLegend(..., gap = unit(4, "mm"), direction = c("vertical", "horizontal"))
```

**Arguments**

...	objects returned by <a href="#">Legend</a>
gap	gap between two legends. The value is a <code>unit</code> object
direction	how to arrange legends

**Value**

A `grob` object

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
lgd1 = Legend(title = "discrete", at = 1:4, labels = letters[1:4],
legend_gp = gpar(fill = 2:5))

require(circlize)
col_fun = colorRamp2(c(-1, 0, 1), c("blue", "white", "red"))
lgd2 = Legend(title = "continuous", at = seq(-1, 1, by = 0.5), col_fun = col_fun)

p1 = packLegend(lgd1, lgd2)
grid.newpage()
grid.draw(p1)

p1 = packLegend(lgd1, lgd2, direction = "horizontal")
grid.newpage()
grid.draw(p1)
```

---

plotDataFrame                      *Quickly visualize a data frame*

---

### Description

Quickly visualize a data frame

### Usage

```
plotDataFrame(df, overlap = 0.25, nlevel = 30, show_row_names = TRUE,
  show_column_names = TRUE, group = NULL, group_names = names(group),
  main_heatmap = NULL, km = 1, split = NULL, cluster_rows = TRUE,
  cluster_columns = TRUE, row_order = NULL, ...)
```

### Arguments

df	a data frame.
overlap	how to group numeric columns. If the overlapping rate between the ranges in the current column and previous numeric column is larger than this value, the two columns are treated as under same measurement and should be grouped.
nlevel	If the number of levels of a character column is larger than this value, the column will be excluded, because it doesn't make any sense to visualize a character vector or matrix that contains huge number of unique elements through a heatmap.
show_row_names	whether show row names after the last heatmap if there are row names.
show_column_names	whether show column names for all heatmaps.
group	a list of index that defines the grouping.
group_names	names for each group.
main_heatmap	which group is the main heatmap?
km	a value larger than 1 means applying k-means clustering on rows for the main heatmap.
split	one or multiple variables that split the rows.
cluster_rows	whether perform clustering on rows of the main heatmap.
cluster_columns	whether perform clustering on columns for all heatmaps.
row_order	order of rows, remember to turn off cluster_rows
...	pass to <a href="#">draw, HeatmapList-method</a> or <a href="#">make_layout, HeatmapList-method</a>

### Details

The data frame contains heterogeneous information. The [plotDataFrame](#) function provides a simple and quick way to visualize information that are stored in a data frame.

There are only a few settings in this function, so the heatmap generated by this function may look ugly (in most of the time). However, users can customize the style of the heatmaps by manually constructing a [HeatmapList](#) object.

**Value**

A [HeatmapList](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
df = data.frame(matrix(rnorm(40), nrow = 10, dimnames = list(letters[1:10], letters[1:4])),
  large = runif(10)*100,
  t1 = sample(letters[1:3], 10, replace = TRUE),
  matrix(runif(60), nrow = 10, dimnames = list(LETTERS[1:10], LETTERS[1:6])),
  t2 = sample(LETTERS[1:3], 10, replace = TRUE))
plotDataFrame(df)
plotDataFrame(df, group = list(1:4, 5, 6, 7:12, 13), group_names = c("mat1", "large", "t1", "mat2", "t2"),
  main_heatmap = 4, km = 2, column_title = "column title", row_title = "row title")
```

---

prepare-Heatmap-method

*Prepare the heatmap*

---

**Description**

Prepare the heatmap

**Usage**

```
## S4 method for signature 'Heatmap'
prepare(object, process_rows = TRUE)
```

**Arguments**

**object** a [Heatmap-class](#) object.  
**process\_rows** whether process rows of the heatmap

**Details**

The preparation of the heatmap includes following steps:

- making clustering on rows if specified (by calling [make\\_row\\_cluster, Heatmap-method](#))
- making clustering on columns if specified (by calling [make\\_column\\_cluster, Heatmap-method](#))
- making the layout of the heatmap (by calling [make\\_layout, Heatmap-method](#))

This function is only for internal use.

**Value**

A [Heatmap-class](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this internal method
NULL
```

---

rowAnnotation	<i>Construct row annotations</i>
---------------	----------------------------------

---

**Description**

Construct row annotations

**Usage**

```
rowAnnotation(...)
```

**Arguments**

... pass to [HeatmapAnnotation](#)

**Details**

The function is identical to

```
HeatmapAnnotation(..., which = "row")
```

**Value**

A [HeatmapAnnotation-class](#) object.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
df = data.frame(type = c("a", "a", "a", "b", "b", "b"))
ha = columnAnnotation(df = df)
```



---

row_anno_barplot	<i>Row annotation which is represented as barplots</i>
------------------	--

---

**Description**

Row annotation which is represented as barplots

**Usage**

```
row_anno_barplot(...)
```

**Arguments**

... pass to [anno\\_barplot](#)

**Details**

A wrapper of [anno\\_barplot](#) with pre-defined which to row.

**Value**

See help page of [anno\\_barplot](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example  
NULL
```

---

row_anno_boxplot	<i>Row annotation which is represented as boxplots</i>
------------------	--

---

**Description**

Row annotation which is represented as boxplots

**Usage**

```
row_anno_boxplot(...)
```

**Arguments**

... pass to [anno\\_boxplot](#)

**Details**

A wrapper of [anno\\_boxplot](#) with pre-defined which to row.

**Value**

See help page of [anno\\_boxplot](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example  
NULL
```

---

row_anno_density	<i>Row annotation which is represented as density plot</i>
------------------	--

---

**Description**

Row annotation which is represented as density plot

**Usage**

```
row_anno_density(...)
```

**Arguments**

```
...          pass to anno\_density
```

**Details**

A wrapper of [anno\\_density](#) with pre-defined which to row.

**Value**

See help page of [anno\\_density](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example  
NULL
```

---

row_anno_histogram	<i>Row annotation which is represented as histogram</i>
--------------------	---

---

**Description**

Row annotation which is represented as histogram

**Usage**

```
row_anno_histogram(...)
```

**Arguments**

... pass to [anno\\_histogram](#)

**Details**

A wrapper of [anno\\_histogram](#) with pre-defined which to row.

**Value**

See help page of [anno\\_histogram](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example
NULL
```

---

row_anno_link	<i>Column annotation which is represented as links</i>
---------------	--

---

**Description**

Column annotation which is represented as links

**Usage**

```
row_anno_link(...)
```

**Arguments**

... pass to [anno\\_link](#)

**Details**

A wrapper of [anno\\_link](#) with pre-defined which to row.

**Value**

See help page of [anno\\_link](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example
NULL
```

---

row_anno_points	<i>Row annotation which is represented as points</i>
-----------------	--

---

**Description**

Row annotation which is represented as points

**Usage**

```
row_anno_points(...)
```

**Arguments**

```
...          pass to anno\_points
```

**Details**

A wrapper of [anno\\_points](#) with pre-defined which to row.

**Value**

See help page of [anno\\_points](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example
NULL
```

---

row_anno_text	<i>Row annotation which is represented as text</i>
---------------	--

---

**Description**

Row annotation which is represented as text

**Usage**

```
row_anno_text(...)
```

**Arguments**

... pass to [anno\\_text](#)

**Details**

A wrapper of [anno\\_text](#) with pre-defined which to row.

**Value**

See help page of [anno\\_text](#)

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example
NULL
```

---

row_dend-dispatch	<i>Method dispatch page for row_dend</i>
-------------------	--

---

**Description**

Method dispatch page for row\_dend.

**Dispatch**

row\_dend can be dispatched on following classes:

- [row\\_dend, HeatmapList-method, HeatmapList-class](#) class method
- [row\\_dend, Heatmap-method, Heatmap-class](#) class method

**Examples**

```
# no example
NULL
```

---

row\_dend-Heatmap-method

*Get row dendrograms from a heatmap*

---

### Description

Get row dendrograms from a heatmap

### Usage

```
## S4 method for signature 'Heatmap'  
row_dend(object)
```

### Arguments

object            a [Heatmap-class](#) object

### Value

A list of dendrograms for which each dendrogram corresponds to a row slice

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
mat = matrix(rnorm(100), 10)  
ht = Heatmap(mat)  
row_dend(ht)  
ht = Heatmap(mat, km = 2)  
row_dend(ht)
```

---

row\_dend-HeatmapList-method

*Get row dendrograms from a heatmap list*

---

### Description

Get row dendrograms from a heatmap list

### Usage

```
## S4 method for signature 'HeatmapList'  
row_dend(object)
```

### Arguments

object            a [HeatmapList-class](#) object

**Value**

A list of dendrograms for which each dendrogram corresponds to a row slice

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
mat = matrix(rnorm(100), 10)
ht_list = Heatmap(mat) + Heatmap(mat)
row_dend(ht_list)
ht_list = Heatmap(mat, km = 2) + Heatmap(mat)
row_dend(ht_list)
```

---

row\_order-dispatch      *Method dispatch page for row\_order*

---

**Description**

Method dispatch page for row\_order.

**Dispatch**

row\_order can be dispatched on following classes:

- [row\\_order, HeatmapList-method, HeatmapList-class](#) class method
- [row\\_order, Heatmap-method, Heatmap-class](#) class method

**Examples**

```
# no example
NULL
```

---

row\_order-Heatmap-method  
*Get row order from a heatmap*

---

**Description**

Get row order from a heatmap

**Usage**

```
## S4 method for signature 'Heatmap'
row_order(object)
```

**Arguments**

object            a [Heatmap-class](#) object

**Value**

A list contains row orders which correspond to the original matrix

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
mat = matrix(rnorm(100), 10)
ht = Heatmap(mat)
row_order(ht)
ht = Heatmap(mat, km = 2)
row_order(ht)
```

---

row\_order-HeatmapList-method

*Get row order from a heatmap list*

---

**Description**

Get row order from a heatmap list

**Usage**

```
## S4 method for signature 'HeatmapList'
row_order(object)
```

**Arguments**

object            a [HeatmapList-class](#) object

**Value**

A list contains row orders which correspond to the original matrix

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
mat = matrix(rnorm(100), 10)
ht_list = Heatmap(mat) + Heatmap(mat)
row_order(ht_list)
ht = Heatmap(mat, km = 2) + Heatmap(mat)
row_order(ht_list)
```



---

selectArea	<i>Select an area in the heatmap</i>
------------	--------------------------------------

---

**Description**

Select an area in the heatmap

**Usage**

```
selectArea(mark = TRUE)
```

**Arguments**

mark                    whether mark the selected area as a rectangle

**Details**

Users can use mouse to click two positions on the heatmap, the function will return the row index and column index for the selected region in the selected matrix.

This function only works under interactive graphical environment.

**Value**

A list containing row index and column index corresponding to the selected region.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# No example for this function
NULL
```

---

set_component_height-Heatmap-method	<i>Set height of each heatmap component</i>
-------------------------------------	---

---

**Description**

Set height of each heatmap component

**Usage**

```
## S4 method for signature 'Heatmap'
set_component_height(object, k, v)
```

**Arguments**

object            a [Heatmap-class](#) object.  
k                which components, see [Heatmap-class](#).  
v                height of the component, a [unit](#) object.

**Details**

This function is only for internal use.

**Value**

This function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# no example for this internal method  
NULL
```

---

```
show-ColorMapping-method  
                          Print ColorMapping object
```

---

**Description**

Print ColorMapping object

**Usage**

```
## S4 method for signature 'ColorMapping'  
show(object)
```

**Arguments**

object            a [ColorMapping-class](#) object.

**Value**

This function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example  
NULL
```

---

show-dispatch	<i>Method dispatch page for show</i>
---------------	--------------------------------------

---

### Description

Method dispatch page for show.

### Dispatch

show can be dispatched on following classes:

- [show, ColorMapping-method, ColorMapping-class](#) class method
- [show, HeatmapAnnotation-method, HeatmapAnnotation-class](#) class method
- [show, SingleAnnotation-method, SingleAnnotation-class](#) class method
- [show, HeatmapList-method, HeatmapList-class](#) class method
- [show, Heatmap-method, Heatmap-class](#) class method

### Examples

```
# no example  
NULL
```

---

show-Heatmap-method	<i>Draw the single heatmap with default parameters</i>
---------------------	--

---

### Description

Draw the single heatmap with default parameters

### Usage

```
## S4 method for signature 'Heatmap'  
show(object)
```

### Arguments

object            a [Heatmap-class](#) object.

### Details

Actually it calls [draw, Heatmap-method](#), but only with default parameters. If users want to customize the heatmap, they can pass parameters directly to [draw, Heatmap-method](#).

### Value

This function returns no value.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
mat = matrix(rnorm(80, 2), 8, 10)
mat = rbind(mat, matrix(rnorm(40, -2), 4, 10))
rownames(mat) = letters[1:12]
colnames(mat) = letters[1:10]

ht = Heatmap(mat)
ht
draw(ht, heatmap_legend_side = "left")
```

---

show-HeatmapAnnotation-method

*Print the Heatmap Annotation object*

---

**Description**

Print the Heatmap Annotation object

**Usage**

```
## S4 method for signature 'HeatmapAnnotation'
show(object)
```

**Arguments**

object            a [HeatmapAnnotation-class](#) object.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example
NULL
```

---

show-HeatmapList-method

*Draw a list of heatmaps with default parameters*

---

### Description

Draw a list of heatmaps with default parameters

### Usage

```
## S4 method for signature 'HeatmapList'  
show(object)
```

### Arguments

object            a [HeatmapList-class](#) object.

### Details

Actually it calls [draw,HeatmapList-method](#), but only with default parameters. If users want to customize the heatmap, they can pass parameters directly to [draw,HeatmapList-method](#).

### Value

This function returns no value.

### Author(s)

Zuguang Gu <z.gu@dkfz.de>

### Examples

```
# There is no example  
NULL
```

---

show-SingleAnnotation-method

*Print the SingleAnnotation object*

---

### Description

Print the SingleAnnotation object

### Usage

```
## S4 method for signature 'SingleAnnotation'  
show(object)
```

**Arguments**

object            a `SingleAnnotation-class` object.

**Value**

No value is returned.

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example
NULL
```

---

SingleAnnotation	<i>Constructor method for SingleAnnotation class</i>
------------------	--

---

**Description**

Constructor method for SingleAnnotation class

**Usage**

```
SingleAnnotation(name, value, col, fun,
  na_col = "grey",
  which = c("column", "row"),
  show_legend = TRUE,
  gp = gpar(col = NA),
  legend_param = list(),
  show_name = FALSE,
  name_gp = gpar(fontsize = 12),
  name_offset = unit(2, "mm"),
  name_side = ifelse(which == "column", "right", "bottom"),
  name_rot = ifelse(which == "column", 0, 90))
```

**Arguments**

name	name for this annotation. If it is not specified, an internal name is assigned.
value	A vector of discrete or continuous annotation.
col	colors corresponding to value. If the mapping is discrete mapping, the value of col should be a vector; If the mapping is continuous mapping, the value of col should be a color mapping function.
fun	a self-defined function to add annotation graphics. The argument of this function should only be a vector of index that corresponds to rows or columns.
na_col	color for NA values in simple annotations.
which	is the annotation a row annotation or a column annotation?

show_legend	if it is a simple annotation, whether show legend when making the complete heatmap.
gp	Since simple annotation is represented as a row of grids. This argument controls graphic parameters for the simple annotation.
legend_param	parameters for the legend. See <a href="#">color_mapping_legend</a> , <a href="#">ColorMapping-method</a> for options.
show_name	whether show annotation name
name_gp	graphic parameters for annotation name
name_offset	offset to the annotation, a <a href="#">unit</a> object
name_side	'right' and 'left' for column annotations and 'top' and 'bottom' for row annotations
name_rot	rotation of the annotation name, can only take values in <code>c(00, 90, 180, 270)</code> .

### Details

The most simple annotation is one row or one column grids in which different colors represent different classes of the data. Here the function use [ColorMapping-class](#) to process such simple annotation. `value` and `col` arguments controls values and colors of the simple annotation and a [ColorMapping-class](#) object will be constructed based on `value` and `col`.

`fun` is used to construct a more complex annotation. Users can add any type of annotation graphics by implementing a function. The only input argument of `fun` is a index of rows or columns which is already adjusted by the clustering. In the package, there are already several annotation graphic function generators: [anno\\_points](#), [anno\\_histogram](#) and [anno\\_boxplot](#).

In the case that row annotations are splitted by rows, `index` corresponding to row orders in each row-slice and `fun` will be applied on each of the row slices.

One thing that users should be careful is the difference of coordinates when the annotation is a row annotation or a column annotation.

### Value

A [SingleAnnotation-class](#) object.

### Author(s)

Zuguang Gu <[z.gu@dkfz.de](mailto:z.gu@dkfz.de)>

### See Also

There are following built-in annotation functions that can be used to generate complex annotations: [anno\\_points](#), [anno\\_barplot](#), [anno\\_histogram](#), [anno\\_boxplot](#), [anno\\_density](#), [anno\\_text](#) and [anno\\_link](#).

### Examples

```
# discrete character
SingleAnnotation(name = "test", value = c("a", "a", "a", "b", "b", "b"))
SingleAnnotation(name = "test", value = c("a", "a", "a", "b", "b", "b"),
  which = "row")

# with defined colors
SingleAnnotation(value = c("a", "a", "a", "b", "b", "b"),
```

```
col = c("a" = "red", "b" = "blue"))

# continuous numbers
require(circlize)
SingleAnnotation(value = 1:10)
SingleAnnotation(value = 1:10, col = colorRamp2(c(1, 10), c("blue", "red")))

# self-defined graphic function
SingleAnnotation(fun = anno_points(1:10))
```

---

SingleAnnotation-class

*Class for a single annotation*

---

## Description

Class for a single annotation

## Details

A complex heatmap always has more than one annotations on rows and columns. Here the [SingleAnnotation-class](#) defines the basic unit of annotations. The most simple annotation is one row or one column grids in which different colors represent different classes of the data. The annotation can also be more complex graphics, such as a boxplot that shows data distribution in corresponding row or column.

The [SingleAnnotation-class](#) is used for storing data for a single annotation and provides methods for drawing annotation graphics.

## Methods

The [SingleAnnotation-class](#) provides following methods:

- [SingleAnnotation](#): constructor method
- [draw,SingleAnnotation-method](#): draw the single annotation.

## Author(s)

Zuguang Gu <z.gu@dkfz.de>

## See Also

The [SingleAnnotation-class](#) is always used internally. The public [HeatmapAnnotation-class](#) contains a list of [SingleAnnotation-class](#) objects and is used to add annotation graphics on heatmaps.

## Examples

```
# for examples, please go to `SingleAnnotation` method page
NULL
```



---

unify_mat_list	<i>Unify a list of matrix</i>
----------------	-------------------------------

---

**Description**

Unify a list of matrix

**Usage**

```
unify_mat_list(mat_list, default = 0)
```

**Arguments**

mat_list	a list of matrix, all of them should have dimension names
default	default values for the newly added rows and columns

**Details**

All matrix will be unified to have same row names and column names

**Value**

A list of matrix

**Author(s)**

Zuguang Gu <z.gu@dkfz.de>

**Examples**

```
# There is no example  
NULL
```

# Index

- [+.AdditiveUnit](#), [5](#)
- [add\\_heatmap](#) ([add\\_heatmap-dispatch](#)), [7](#)
- [add\\_heatmap](#), [Heatmap-method](#)
  - ([add\\_heatmap-Heatmap-method](#)), [7](#)
- [add\\_heatmap](#), [HeatmapAnnotation-method](#)
  - ([add\\_heatmap-HeatmapAnnotation-method](#)), [8](#)
- [add\\_heatmap](#), [HeatmapList-method](#)
  - ([add\\_heatmap-HeatmapList-method](#)), [9](#)
- [add\\_heatmap-dispatch](#), [7](#)
- [add\\_heatmap-Heatmap-method](#), [7](#)
- [add\\_heatmap-HeatmapAnnotation-method](#), [8](#)
- [add\\_heatmap-HeatmapList-method](#), [9](#)
- [AdditiveUnit](#), [6](#)
- [AdditiveUnit-class](#), [6](#)
- [adjust\\_dend\\_by\\_leaf\\_width](#), [10](#), [10](#), [65](#)
- [anno\\_barplot](#), [12](#), [23](#), [24](#), [97](#), [111](#)
- [anno\\_boxplot](#), [13](#), [24](#), [97](#), [98](#), [111](#)
- [anno\\_density](#), [14](#), [25](#), [98](#), [111](#)
- [anno\\_histogram](#), [15](#), [25](#), [26](#), [99](#), [111](#)
- [anno\\_link](#), [16](#), [26](#), [99](#), [100](#), [111](#)
- [anno\\_oncoprint\\_barplot](#), [17](#)
- [anno\\_points](#), [17](#), [27](#), [100](#), [111](#)
- [anno\\_text](#), [18](#), [27](#), [28](#), [89](#), [90](#), [101](#), [111](#)
- [annotation\\_legend\\_size](#)
  - ([annotation\\_legend\\_size-HeatmapList-method](#)), [11](#)
- [annotation\\_legend\\_size](#), [HeatmapList-method](#)
  - ([annotation\\_legend\\_size-HeatmapList-method](#)), [11](#)
- [annotation\\_legend\\_size-HeatmapList-method](#), [11](#)
- [as.dist](#), [46](#)
- [color\\_mapping\\_legend](#)
  - ([color\\_mapping\\_legend-ColorMapping-method](#)), [21](#)
- [color\\_mapping\\_legend](#), [ColorMapping-method](#)
  - ([color\\_mapping\\_legend-ColorMapping-method](#)), [21](#)
- [color\\_mapping\\_legend-ColorMapping-method](#), [21](#)
- [ColorMapping](#), [19](#), [20](#), [66](#)
- [ColorMapping-class](#), [20](#)
- [colorRamp2](#), [19](#), [45](#), [66](#), [67](#)
- [column\\_anno\\_barplot](#), [23](#)
- [column\\_anno\\_boxplot](#), [24](#)
- [column\\_anno\\_density](#), [25](#)
- [column\\_anno\\_histogram](#), [25](#)
- [column\\_anno\\_link](#), [26](#)
- [column\\_anno\\_points](#), [27](#)
- [column\\_anno\\_text](#), [27](#)
- [column\\_dend](#) ([column\\_dend-dispatch](#)), [28](#)
- [column\\_dend](#), [Heatmap-method](#)
  - ([column\\_dend-Heatmap-method](#)), [28](#)
- [column\\_dend](#), [HeatmapList-method](#)
  - ([column\\_dend-HeatmapList-method](#)), [29](#)
- [column\\_dend-dispatch](#), [28](#)
- [column\\_dend-Heatmap-method](#), [28](#)
- [column\\_dend-HeatmapList-method](#), [29](#)
- [column\\_order](#) ([column\\_order-dispatch](#)), [30](#)
- [column\\_order](#), [Heatmap-method](#)
  - ([column\\_order-Heatmap-method](#)), [30](#)
- [column\\_order](#), [HeatmapList-method](#)
  - ([column\\_order-HeatmapList-method](#)), [31](#)
- [column\\_order-dispatch](#), [30](#)
- [column\\_order-Heatmap-method](#), [30](#)
- [column\\_order-HeatmapList-method](#), [31](#)
- [columnAnnotation](#), [23](#), [74](#)
- [ComplexHeatmap-package](#), [4](#)
- [component\\_height](#)
  - ([component\\_height-dispatch](#)), [31](#)
- [component\\_height](#), [Heatmap-method](#)
  - ([component\\_height-Heatmap-method](#)), [32](#)
- [component\\_height](#), [HeatmapList-method](#)
  - ([component\\_height-HeatmapList-method](#)), [33](#)
- [component\\_height-dispatch](#), [31](#)

- component\_height-Heatmap-method, [32](#)
- component\_height-HeatmapList-method, [33](#)
- component\_width
  - (component\_width-dispatch), [33](#)
  - (component\_width,Heatmap-method (component\_width-Heatmap-method), [34](#)
  - (component\_width,HeatmapList-method (component\_width-HeatmapList-method), [35](#)
- component\_width-dispatch, [33](#)
- component\_width-Heatmap-method, [34](#)
- component\_width-HeatmapList-method, [35](#)
- decorate\_annotation, [35](#)
- decorate\_column\_dend, [36](#)
- decorate\_column\_names, [37](#)
- decorate\_column\_title, [38](#)
- decorate\_dend, [36](#), [37](#), [38](#), [41](#)
- decorate\_dimnames, [37](#), [39](#), [42](#)
- decorate\_heatmap\_body, [40](#)
- decorate\_row\_dend, [41](#)
- decorate\_row\_names, [42](#)
- decorate\_row\_title, [43](#)
- decorate\_title, [38](#), [43](#), [43](#)
- dendrogram, [10](#), [64](#), [65](#), [67](#)
- density, [14](#), [45](#)
- densityHeatmap, [44](#)
- dist, [47](#), [67](#)
- dist2, [46](#)
- draw (draw-dispatch), [47](#)
- draw,Heatmap-method
  - (draw-Heatmap-method), [48](#)
- draw,HeatmapAnnotation-method
  - (draw-HeatmapAnnotation-method), [49](#)
- draw,HeatmapList-method
  - (draw-HeatmapList-method), [50](#)
- draw,SingleAnnotation-method
  - (draw-SingleAnnotation-method), [52](#)
- draw-dispatch, [47](#)
- draw-Heatmap-method, [48](#)
- draw-HeatmapAnnotation-method, [49](#)
- draw-HeatmapList-method, [50](#)
- draw-SingleAnnotation-method, [52](#)
- draw\_annotation
  - (draw\_annotation-Heatmap-method), [53](#)
- draw\_annotation,Heatmap-method
  - (draw\_annotation-Heatmap-method), [53](#)
- draw\_annotation-Heatmap-method, [53](#)
- draw\_annotation\_legend
  - (draw\_annotation\_legend-HeatmapList-method), [54](#)
- draw\_annotation\_legend,HeatmapList-method
  - (draw\_annotation\_legend-HeatmapList-method), [54](#)
- draw\_annotation\_legend-HeatmapList-method, [54](#)
- draw\_dend (draw\_dend-Heatmap-method), [54](#)
- draw\_dend,Heatmap-method
  - (draw\_dend-Heatmap-method), [54](#)
- draw\_dend-Heatmap-method, [54](#)
- draw\_dimnames
  - (draw\_dimnames-Heatmap-method), [55](#)
- draw\_dimnames,Heatmap-method
  - (draw\_dimnames-Heatmap-method), [55](#)
- draw\_dimnames-Heatmap-method, [55](#)
- draw\_heatmap\_body
  - (draw\_heatmap\_body-Heatmap-method), [56](#)
- draw\_heatmap\_body,Heatmap-method
  - (draw\_heatmap\_body-Heatmap-method), [56](#)
- draw\_heatmap\_body-Heatmap-method, [56](#)
- draw\_heatmap\_body-Heatmap-method, [56](#)
- draw\_heatmap\_legend
  - (draw\_heatmap\_legend-HeatmapList-method), [57](#)
- draw\_heatmap\_legend,HeatmapList-method
  - (draw\_heatmap\_legend-HeatmapList-method), [57](#)
- draw\_heatmap\_legend-HeatmapList-method, [57](#)
- draw\_heatmap\_list
  - (draw\_heatmap\_list-HeatmapList-method), [58](#)
- draw\_heatmap\_list,HeatmapList-method
  - (draw\_heatmap\_list-HeatmapList-method), [58](#)
- draw\_heatmap\_list-HeatmapList-method, [58](#)
- draw\_title (draw\_title-dispatch), [59](#)
- draw\_title,Heatmap-method
  - (draw\_title-Heatmap-method), [59](#)
- draw\_title,HeatmapList-method
  - (draw\_title-HeatmapList-method), [60](#)
- draw\_title-dispatch, [59](#)
- draw\_title-Heatmap-method, [59](#)
- draw\_title-HeatmapList-method, [60](#)



row\_dend-HeatmapList-method, 102  
row\_order (row\_order-dispatch), 103  
row\_order, Heatmap-method  
    (row\_order-Heatmap-method), 103  
row\_order, HeatmapList-method  
    (row\_order-HeatmapList-method),  
    104  
row\_order-dispatch, 103  
row\_order-Heatmap-method, 103  
row\_order-HeatmapList-method, 104  
rowAnnotation, 74, 96  
  
seekViewport, 36, 39–41, 44  
selectArea, 105  
set\_component\_height  
    (set\_component\_height-Heatmap-method),  
    105  
set\_component\_height, Heatmap-method  
    (set\_component\_height-Heatmap-method),  
    105  
set\_component\_height-Heatmap-method,  
    105  
show (show-dispatch), 107  
show, ColorMapping-method  
    (show-ColorMapping-method), 106  
show, Heatmap-method  
    (show-Heatmap-method), 107  
show, HeatmapAnnotation-method  
    (show-HeatmapAnnotation-method),  
    108  
show, HeatmapList-method  
    (show-HeatmapList-method), 109  
show, SingleAnnotation-method  
    (show-SingleAnnotation-method),  
    109  
show-ColorMapping-method, 106  
show-dispatch, 107  
show-Heatmap-method, 107  
show-HeatmapAnnotation-method, 108  
show-HeatmapList-method, 109  
show-SingleAnnotation-method, 109  
SingleAnnotation, 73, 79, 110, 112  
SingleAnnotation-class, 112  
  
textGrob, 89, 90  
  
unify\_mat\_list, 91, 113  
unit, 11, 16, 18, 32–35, 51, 67–69, 74, 78, 80,  
    81, 86, 89–93, 106, 111  
  
viewport, 22, 49, 55, 56, 59, 64