Package 'ComplexHeatmap'

April 12, 2022

Type Package

Title Make Complex Heatmaps

Version 2.10.0

Date 2021-10-17

Author Zuguang Gu

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

Depends R (>= 3.5.0), methods, grid, graphics, stats, grDevices

Imports circlize (>= 0.4.5), GetoptLong, colorspace, clue, RColorBrewer, GlobalOptions (>= 0.1.0), png, digest, IRanges, matrixStats, foreach, doParallel

Suggests testthat (>= 1.0.0), knitr, markdown, dendsort, jpeg, tiff, fastcluster, EnrichedHeatmap, dendextend (>= 1.0.1), grImport, grImport2, glue, GenomicRanges, gridtext, pheatmap (>= 1.0.12), gridGraphics, gplots, rmarkdown, Cairo

VignetteBuilder knitr

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

biocViews Software, Visualization, Sequencing

URL `https://github.com/jokergoo/ComplexHeatmap`,
`https://jokergoo.github.io/ComplexHeatmap-reference/book/`

License MIT + file LICENSE

git_url https://git.bioconductor.org/packages/ComplexHeatmap

git_branch RELEASE_3_14

git_last_commit 170df82

git_last_commit_date 2021-10-26

Date/Publication 2022-04-12

R topics documented:

`ComplexHeatmap-package` *Make complex heatmaps*

Description

Make complex heatmaps

Details

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

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

- `Heatmap-class`: a single heatmap containing heatmap body, row/column names, titles, dendrograms and annotations.
- `HeatmapList-class`: a list of heatmaps and annotations.
- `HeatmapAnnotation-class`: a list of row/column annotations.

There are also several internal classes:

- `SingleAnnotation-class`: a single row annotation or column annotation.
- `ColorMapping-class`: mapping from values to colors.
- `AnnotationFunction-class`: construct an annotation function which allows subsetting.

Following two high-level functions take use of functionality of complex heatmaps:

- `oncoPrint`: oncoPrint plot which visualize genomic alterations in a set of genes.
- `densityHeatmap`: use heatmaps to visualize density distributions.

The complete reference of ComplexHeatmap package is available at `http://jokergoo.github.io/ComplexHeatmap-reference/book.`

Examples

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

`+.AdditiveUnit` *Horizontally Add Heatmaps or Annotations to a Heatmap List*

Description

Horizontally Add Heatmaps or 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. Column annotations should be added to the heatmap list by `%v%`.

`x` and `y` can also be `NULL`.

Value

A `HeatmapList-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`%v%` operator is used for vertical heatmap list.

Examples

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

AdditiveUnit *Constructor Method for AdditiveUnit Class*

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

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

AdditiveUnit-class *Class for Concatenating Heatmaps and Annotations*

Description

Class for Concatenating Heatmaps and Annotations

Details

This class is a super class for `Heatmap-class`, `HeatmapList-class` and `HeatmapAnnotation-class` classes. It is only designed for `+` generic method and the `%v%v` method so that above three classes can be appended to each other.

Examples

```
# There is 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,Heatmap-method`, `Heatmap-class` class method
- `add_heatmap,HeatmapList-method`, `HeatmapList-class` class method
- `add_heatmap,HeatmapAnnotation-method`, `HeatmapAnnotation-class` class method

Examples

```
# no example
NULL
```

---

add_heatmap-Heatmap-method *Add Heatmap to the Heatmap List*

Description

Add Heatmap to the Heatmap List

Usage

```
## S4 method for signature 'Heatmap'
add_heatmap(object, x, direction = c("horizontal", "vertical"))
```

Arguments

`object`	A `Heatmap-class` object.
`x`	a `Heatmap-class` object, a `HeatmapAnnotation-class` object or a `HeatmapList-class` object.
`direction`	Whether the heatmap is added horizontal or vertically?

Details

Normally we directly use + for horizontal concatenation and `%v%` for vertical concatenation.

Value

A `HeatmapList-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

---

add_heatmap-HeatmapAnnotation-method
Add Annotations or Heatmaps as a Heatmap List

Description

Add Annotations or Heatmaps as a Heatmap List

Usage

```
## S4 method for signature 'HeatmapAnnotation'
add_heatmap(object, x, direction = c("horizontal", "vertical"))
```

Arguments

`object`	A `HeatmapAnnotation-class` object.
`x`	A `Heatmap-class` object, a `HeatmapAnnotation-class` object or a `HeatmapList-class` object.
`direction`	Whether it is horizontal list or a vertical list?

Details

Normally we directly use + for horizontal concatenation and `%v%` for vertical concatenation.

Value

A `HeatmapList-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

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, direction = c("horizontal", "vertical"))
```

Arguments

`object`	a `HeatmapList-class` object.
`x`	a `Heatmap-class` object or a `HeatmapAnnotation-class` object or a `HeatmapList-class` object.
`direction`	direction of the concatenation.

Details

There is a shortcut function `+.AdditiveUnit`.

Value

A `HeatmapList-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

adjust_dend_by_x *Adjust the Positions of nodes/leaves in the Dendrogram*

Description

Adjust the Positions of nodes/leaves in the Dendrogram

Usage

```
adjust_dend_by_x(dend, leaf_pos = 1:nobs(dend)-0.5)
```

Arguments

dend	A `dendrogram` object.
leaf_pos	A vector of positions of leaves. The value can also be a `unit` object.

Details

The positions of nodes stored as `x` attribute are recalculated based on the new positions of leaves.

By default, the position of leaves are at 0.5, 1.5, ..., n-0.5.

Examples

```
m = matrix(rnorm(100), 10)
dend = as.dendrogram(hclust(dist(m)))
dend = adjust_dend_by_x(dend, sort(runif(10)))
str(dend)
dend = adjust_dend_by_x(dend, unit(1:10, "cm"))
str(dend)
```

adjust_heatmap_list-HeatmapList-method *Adjust Heatmap List*

Description

Adjust Heatmap List

Usage

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

Arguments

object	A `HeatmapList-class` object.

Details

This function adjusts settings in all other heatmaps according to the main heatmap. It also adjust the size of heatmap annotations to make them aligned nicely.

This function is only for internal use.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

`alter_graphic` *Automatically generate alter_fun*

Description

Automatically generate alter_fun

Usage

```
alter_graphic(graphic = c("rect", "point"),
    width = 1, height = 1,
    horiz_margin = unit(1, "pt"), vertical_margin = unit(1, "pt"),
    fill = "red", col = NA, pch = 16, ...)
```

Arguments

`graphic`	Graphic to draw.
`width`	Relative width of the rectangle.
`height`	Relative height of the rectangle.
`horiz_margin`	Horizontal margin. E.g. if you want 1mm margin on top and 1mm margin at bottom of the rectangle, set this value to `unit(1,'mm')`.
`vertical_margin`	Vertical margin.
`fill`	Filled color.
`col`	Border color.
`pch`	Pch for points
`...`	Pass to `gpar`

Details

This function aims to simplify the definition of functions in `alter_fun`. Now it only supports rectangles and points.

Examples

```
mat = read.table(textConnection(
"s1,s2,s3
g1,snv;indel,snv,indel
g2,,snv;indel,snv
g3,snv,,indel;snv"), row.names = 1, header = TRUE, sep = ",", stringsAsFactors = FALSE)
mat = as.matrix(mat)
col = c(snv = "red", indel = "blue")

oncoPrint(mat,
alter_fun = list(
snv = alter_graphic("rect", width = 0.9, height = 0.9, fill = col["snv"]),
indel = alter_graphic("rect", width = 0.9, height = 0.9, fill = col["indel"])
), col = col)
```

`AnnotationFunction` *Constructor of AnnotationFunction Class*

Description

Constructor of AnnotationFunction Class

Usage

```
AnnotationFunction(fun, fun_name = "", which = c("column", "row"), cell_fun = NULL,
    var_import = list(), n = NA, data_scale = c(0, 1), subset_rule = list(),
   subsetable = length(subset_rule) > 0, show_name = TRUE, width = NULL, height = NULL)
```

Arguments

`fun`	A function which defines how to draw the annotation. See **Details** section.
`fun_name`	The name of the function. It is only used for printing the object.
`which`	Whether it is drawn as a column annotation or a row annotation?
`cell_fun`	A simplified version of `fun`. `cell_fun` only accepts one single index and it draws repeatedly in each annotation cell.
`var_import`	The names of the variables or the variable themselves that the annotation function depends on. See **Details** section.
`n`	Number of observations in the annotation. It is not mandatory, but it is better to provide this information so that the higher order `HeatmapAnnotation` knows it and it can perform check on the consistency of annotations and heatmaps.

`data_scale`	The data scale on the data axis (y-axis for column annotation and x-axis for row annotation). It is only used when `decorate_annotation` is used with "native" unit coordinates.
`subset_rule`	The rule of subsetting variables in `var_import`. It should be set when users want the final object to be subsetable. See **Details** section.
`subsetable`	Whether the object is subsetable?
`show_name`	It is used to turn off the drawing of annotation names in `HeatmapAnnotation`. Annotations always have names associated and normally they will be drawn beside the annotation graphics to tell what the annotation is about. e.g. the annotation names put beside the points annotation graphics. However, for some of the annotations, the names are not necessarily to be drawn, such as text annotations drawn by `anno_text` or an empty annotation drawn by `anno_empty`. In this case, when `show_names` is set to `FALSE`, there will be no annotation names drawn for the annotation.
`width`	The width of the plotting region (the viewport) that the annotation is drawn. If it is a row annotation, the width must be an absolute unit. Since the `AnnotationFunction` object is always contained by the `SingleAnnotation-class`object, you can only set the width of row annotations or height of column annotations, while e.g. the height of the row annotation is always `unit(1,"npc")` which means it always fully filled in the parent `SingleAnnotation` and only in `SingleAnnotation` or even `HeatmapAnnotation` can adjust the height of the row annotations.
`height`	The height of the plotting region (the viewport) that the annotation is drawn. If it is a column annotation, the width must be an absolute unit.

Details

In the package, we have implemted quite a lot annotation functions by `AnnotationFunction` constructor: `anno_empty`, `anno_image`, `anno_points`, `anno_lines`, `anno_barplot`, `anno_boxplot`, `anno_histogram`, `anno_density`, `anno_joyplot`, `anno_horizon`, `anno_text` and `anno_mark`. These built-in annotation functions support as both row annotations and column annotations and they are are all subsettable.

The build-in annotation functions are already enough for most of the analysis, nevertheless, if users want to know more about how to construct the AnnotationFunction class manually, they can refer to `https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#implement-new-annotation-functions`.

Value

A `AnnotationFunction-class` object which can be used in `HeatmapAnnotation`.

Examples

```
x = 1:10
anno1 = AnnotationFunction(
    fun = function(index, k, n) {
        n = length(index)
        pushViewport(viewport(xscale = c(0.5, n + 0.5), yscale = c(0, 10)))
        grid.rect()
```

```
            grid.points(1:n, x[index], default.units = "native")
            if(k == 1) grid.yaxis()
            popViewport()
        },
        var_import = list(x = x),
        n = 10,
        subsetable = TRUE,
        height = unit(2, "cm")
    )
    m = rbind(1:10, 11:20)
    Heatmap(m, top_annotation = HeatmapAnnotation(foo = anno1))
    Heatmap(m, top_annotation = HeatmapAnnotation(foo = anno1), column_km = 2)
```

`AnnotationFunction-class` *The AnnotationFunction Class*

Description

The AnnotationFunction Class

Details

The heatmap annotation is basically graphics aligned to the heatmap columns or rows. There is no restriction for the graphic types, e.g. it can be heatmap-like annotation or points. Here the AnnotationFunction class is designed for creating complex and flexible annotation graphics. As the main part of the class, it uses a user-defined function to define the graphics. It also keeps information of the size of the plotting regions of the annotation. And most importantly, it allows subsetting to the annotation to draw a subset of the graphics, which is the base for the splitting of the annotations.

See `AnnotationFunction` constructor for details.

Examples

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

`annotation_axis_grob` *Grob for Annotation Axis*

Description

Grob for Annotation Axis

Usage

```
annotation_axis_grob(at = NULL, labels = at, labels_rot = 0, gp = gpar(),
    side = "left", facing = "outside", direction = "normal", scale = NULL)
```

Arguments

`at`	Break values. If it is not specified, it is inferred from data scale in current viewport.
`labels`	Corresponding labels.
`labels_rot`	Rotations of labels.
`gp`	Graphic parameters.
`side`	side of the axis of the annotation viewport.
`facing`	Facing of the axis.
`direction`	Direction of the axis. Value should be "normal" or "reverse".
`scale`	The data scale. If it is `NULL`, it is inferred from current viewport.

Value

A `grob` object.

Examples

```
gb = annotation_axis_grob(at = 1:5, labels = month.name[1:5], labels_rot = 0,
    side = "left", facing = "outside")
grid.newpage()
pushViewport(viewport(xscale = c(0, 4), yscale = c(0, 6), width = 0.6, height = 0.6))
grid.rect()
grid.text('side = "left", facing = "outside"')
grid.draw(gb)
popViewport()

gb = annotation_axis_grob(at = 1:5, labels = month.name[1:5], labels_rot = 0,
    side = "left", facing = "inside")
grid.newpage()
pushViewport(viewport(xscale = c(0, 4), yscale = c(0, 6), width = 0.6, height = 0.6))
grid.rect()
grid.text('side = "left", facing = "inside"')
grid.draw(gb)
popViewport()

gb = annotation_axis_grob(at = 1:5, labels = month.name[1:5], labels_rot = 0,
    side = "right", facing = "outside")
grid.newpage()
pushViewport(viewport(xscale = c(0, 4), yscale = c(0, 6), width = 0.6, height = 0.6))
grid.rect()
grid.text('side = "right", facing = "outside"')
grid.draw(gb)
popViewport()

gb = annotation_axis_grob(at = 1:5, labels = month.name[1:5], labels_rot = 0,
    side = "right", facing = "inside")
grid.newpage()
pushViewport(viewport(xscale = c(0, 4), yscale = c(0, 6), width = 0.6, height = 0.6))
grid.rect()
grid.text('side = "right", facing = "inside"')
grid.draw(gb)
popViewport()

gb = annotation_axis_grob(at = 1:3, labels = month.name[1:3], labels_rot = 0,
    side = "top", facing = "outside")
grid.newpage()
pushViewport(viewport(xscale = c(0, 4), yscale = c(0, 6), width = 0.6, height = 0.6))
grid.rect()
grid.text('side = "top", facing = "outside"')
grid.draw(gb)
popViewport()

gb = annotation_axis_grob(at = 1:3, labels = month.name[1:3], labels_rot = 90,
    side = "top", facing = "outside")
grid.newpage()
pushViewport(viewport(xscale = c(0, 4), yscale = c(0, 6), width = 0.6, height = 0.6))
grid.rect()
grid.text('side = "top", facing = "outside"')
grid.draw(gb)
popViewport()

gb = annotation_axis_grob(at = 1:3, labels = month.name[1:3], labels_rot = 45,
    side = "top", facing = "outside")
grid.newpage()
pushViewport(viewport(xscale = c(0, 4), yscale = c(0, 6), width = 0.6, height = 0.6))
grid.rect()
grid.text('side = "top", facing = "outside"')
grid.draw(gb)
popViewport()

gb = annotation_axis_grob(at = 1:3, labels = month.name[1:3], labels_rot = 0,
    side = "top", facing = "inside")
grid.newpage()
pushViewport(viewport(xscale = c(0, 4), yscale = c(0, 6), width = 0.6, height = 0.6))
grid.rect()
grid.text('side = "top", facing = "inside"')
grid.draw(gb)
popViewport()

gb = annotation_axis_grob(at = 1:3, labels = month.name[1:3], labels_rot = 0,
    side = "bottom", facing = "outside")
grid.newpage()
pushViewport(viewport(xscale = c(0, 4), yscale = c(0, 6), width = 0.6, height = 0.6))
grid.rect()
grid.text('side = "bottom", facing = "outside"')
grid.draw(gb)
popViewport()

gb = annotation_axis_grob(at = 1:3, labels = month.name[1:3], labels_rot = 0,
    side = "bottom", facing = "inside")
grid.newpage()
pushViewport(viewport(xscale = c(0, 4), yscale = c(0, 6), width = 0.6, height = 0.6))
```

```
grid.rect()
grid.text('side = "bottom", facing = "inside"')
grid.draw(gb)
popViewport()

grid.newpage()
pushViewport(viewport(xscale = c(0, 4), yscale = c(0, 6), width = 0.6, height = 0.6))
gb = annotation_axis_grob(labels_rot = 0, side = "left", facing = "outside")
grid.rect()
grid.text('side = "left", facing = "outside"')
grid.draw(gb)
popViewport()

grid.newpage()
pushViewport(viewport(xscale = c(0, 4), yscale = c(0, 6), width = 0.6, height = 0.6))
gb = annotation_axis_grob(side = "left", direction = "reverse")
grid.rect()
grid.text('side = "left", direction = "reverse')
grid.draw(gb)
popViewport()

grid.newpage()
pushViewport(viewport(xscale = c(0, 4), yscale = c(0, 6), width = 0.6, height = 0.6))
gb = annotation_axis_grob(side = "bottom", direction = "reverse")
grid.rect()
grid.text('side = "bottom", directio = "reverse"')
grid.draw(gb)
popViewport()
```

`annotation_legend_size-HeatmapList-method`

Size of the Annotation Legends

Description

Size of the Annotation Legends

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. It is normally constructed by `Legend`.
`...`	Other arguments.

Details

Internally, all annotation legends are packed by `packLegend` as a single `grob` object.

This function is only for internal use.

Value

A `unit` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

`anno_barplot` *Barplot Annotation*

Description

Barplot Annotation

Usage

```
anno_barplot(x, baseline = 0, which = c("column", "row"), border = TRUE, bar_width = 0.6, beside = FALSE
    gp = gpar(fill = "#CCCCCC"), ylim = NULL, extend = 0.05, axis = TRUE,
    axis_param = default_axis_param(which),
    add_numbers = FALSE, numbers_gp = gpar(fontsize = 8),
   numbers_rot = ifelse(which == "column", 45, 0), numbers_offset = unit(2, "mm"),
    width = NULL, height = NULL, ...)
```

Arguments

`x`	The value vector. The value can be a vector or a matrix. The length of the vector or the number of rows of the matrix is taken as the number of the observations of the annotation. If `x` is a vector, the barplots will be represented as stacked barplots.
`baseline`	baseline of bars. The value should be "min" or "max", or a numeric value. It is enforced to be zero for stacked barplots.
`which`	Whether it is a column annotation or a row annotation?
`border`	Wether draw borders of the annotation region?
`bar_width`	Relative width of the bars. The value should be smaller than one.
`beside`	When `x` is a matrix, will bars be positioned beside each other or as stacked bars?

gp	Graphic parameters for bars. The length of each graphic parameter can be 1, length of x if x is a vector, or number of columns of x is x is a matrix.
ylim	Data ranges. By default it is range(x) if x is a vector, or range(rowSums(x)) if x is a matrix.
extend	The extension to both side of ylim. The value is a percent value corresponding to ylim[2] -ylim[1].
axis	Whether to add axis?
axis_param	parameters for controlling axis. See default_axis_param for all possible settings and default parameters.
add_numbers	Whether to add numbers to the bars. It only works when x is a simple vector.
numbers_gp	Graphics parameters for the numbers.
numbers_rot	Rotation of numbers.
numbers_offset	Offset to the default positions (1mm away the top of the bars).
width	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
height	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.
...	Other arguments.

Value

An annotation function which can be used in HeatmapAnnotation.

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#barplot_annotation

Examples

```
anno = anno_barplot(1:10)
draw(anno, test = "a vector")

m = matrix(runif(4*10), nc = 4)
m = t(apply(m, 1, function(x) x/sum(x)))
anno = anno_barplot(m, gp = gpar(fill = 2:5), bar_width = 1, height = unit(6, "cm"))
draw(anno, test = "proportion matrix")
```

anno_block *Block annotation*

Description

Block annotation

Usage

```
anno_block(gp = gpar(), labels = NULL, labels_gp = gpar(),
    labels_rot = ifelse(which == "row", 90, 0),
    labels_offset = unit(0.5, "npc"), labels_just = "center",
    which = c("column", "row"), width = NULL, height = NULL, show_name = FALSE,
    graphics = NULL)
```

Arguments

`gp`	Graphic parameters.
`labels`	Labels put on blocks.
`labels_gp`	Graphic parameters for labels.
`labels_rot`	Rotation for labels.
`labels_offset`	Positions of the labels. It controls offset on y-directions for column annotation and on x-directoin for row annotation.
`labels_just`	Jusification of the labels.
`which`	Is it a row annotation or a column annotation?
`width`	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
`height`	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.
`show_name`	Whether show annotatio name.
`graphics`	A self-defined function that draws graphics in each slice. It must have two arguments: 1. row/column indices for the current slice and 2. a vector of levels from the split variable that correspond to current slice. When `graphics` is set, all other graphics parameters in `anno_block` are ignored.

Details

The block annotation is used for representing slices. The length of all arguments should be 1 or the number of slices.

Value

An annotation function which can be used in `HeatmapAnnotation`.

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#block-annotation

Examples

```
Heatmap(matrix(rnorm(100), 10),
    top_annotation = HeatmapAnnotation(foo = anno_block(gp = gpar(fill = 2:4),
        labels = c("group1", "group2", "group3"), labels_gp = gpar(col = "white"))),
    column_km = 3,
    left_annotation = rowAnnotation(foo = anno_block(gp = gpar(fill = 2:4),
        labels = c("group1", "group2", "group3"), labels_gp = gpar(col = "white"))),
    row_km = 3)

# =============  set the graphics argument ==============
col = c("1" = "red", "2" = "blue", "A" = "green", "B" = "orange")
Heatmap(matrix(rnorm(100), 10), row_km = 2, row_split = sample(c("A", "B"), 10, replace = TRUE)) +
rowAnnotation(foo = anno_block(
graphics = function(index, levels) {
grid.rect(gp = gpar(fill = col[levels[2]], col = "black"))
grid.text(paste(levels, collapse = ","), 0.5, 0.5, rot = 90,
gp = gpar(col = col[levels[1]]))
}
))

labels = c("1" = "one", "2" = "two", "A" = "Group_A", "B" = "Group_B")
Heatmap(matrix(rnorm(100), 10), row_km = 2, row_split = sample(c("A", "B"), 10, replace = TRUE)) +
rowAnnotation(foo = anno_block(graphics = function(index, levels) {
grid.rect(gp = gpar(fill = col[levels[2]], col = "black"))
grid.text(paste(labels[levels], collapse = ","), 0.5, 0.5, rot = 90,
gp = gpar(col = col[levels[1]]))
}))

Heatmap(matrix(rnorm(100), 10), row_km = 2, row_split = sample(c("A", "B"), 10, replace = TRUE)) +
rowAnnotation(foo = anno_block(
graphics = function(index, levels) {
grid.rect(gp = gpar(fill = col[levels[2]], col = "black"))
txt = paste(levels, collapse = ",")
txt = paste0(txt, "\n", length(index), " rows")
grid.text(txt, 0.5, 0.5, rot = 0,
gp = gpar(col = col[levels[1]]))
},
width = unit(3, "cm")
))
```

---

anno_boxplot *Boxplot Annotation*

Description

Boxplot Annotation

Usage

```
anno_boxplot(x, which = c("column", "row"), border = TRUE,
   gp = gpar(fill = "#CCCCCC"), ylim = NULL, extend = 0.05, outline = TRUE, box_width = 0.6,
   pch = 1, size = unit(2, "mm"), axis = TRUE, axis_param = default_axis_param(which),
    width = NULL, height = NULL, ...)
```

Arguments

`x`	A matrix or a list. If `x` is a matrix and if `which` is `column`, statistics for boxplots are calculated by columns, if `which` is `row`, the calculation is done by rows.
`which`	Whether it is a column annotation or a row annotation?
`border`	Wether draw borders of the annotation region?
`gp`	Graphic parameters for the boxes. The length of the graphic parameters should be one or the number of observations.
`ylim`	Data ranges.
`extend`	The extension to both side of `ylim`. The value is a percent value corresponding to `ylim[2] -ylim[1]`.
`outline`	Whether draw outline of boxplots?
`box_width`	Relative width of boxes. The value should be smaller than one.
`pch`	Point style.
`size`	Point size.
`axis`	Whether to add axis?
`axis_param`	parameters for controlling axis. See `default_axis_param` for all possible settings and default parameters.
`width`	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
`height`	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.
`...`	Other arguments.

Value

An annotation function which can be used in `HeatmapAnnotation`.

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#box-annotation

Examples

```
set.seed(123)
m = matrix(rnorm(100), 10)
anno = anno_boxplot(m, height = unit(4, "cm"))
draw(anno, test = "anno_boxplot")
anno = anno_boxplot(m, height = unit(4, "cm"), gp = gpar(fill = 1:10))
draw(anno, test = "anno_boxplot with gp")
```

anno_customize *Customized annotation*

Description

Customized annotation

Usage

```
anno_customize(x, graphics = list(), which = c("column", "row"),
    border = TRUE, width = NULL, height = NULL, verbose = TRUE)
```

Arguments

x	A categorical variable.
graphics	A list of functions that define graphics for each level in x.
which	Is it a row annotation or a column annotation?
width	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
height	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.
border	Whether to draw border.
verbose	Whether to print messages.

Details

Functions in graphics define simple graphics drawn in each annotation cell. The function takes four arguments:

x,y Center of the annotation cell.

w,h Width and height of the annotation cell.

Value

An annotation function which can be used in HeatmapAnnotation.

Examples

```
x = sort(sample(letters[1:3], 10, replace = TRUE))
graphics = list(
    "a" = function(x, y, w, h) grid.points(x, y, pch = 16),
    "b" = function(x, y, w, h) grid.rect(x, y, w*0.8, h*0.8, gp = gpar(fill = "red")),
   "c" = function(x, y, w, h) grid.segments(x - 0.5*w, y - 0.5*h, x + 0.5*w, y + 0.5*h, gp = gpar(lty = 2))
)

anno = anno_customize(x, graphics = graphics)
```

```
m = matrix(rnorm(100), 10)
Heatmap(m, top_annotation = HeatmapAnnotation(bar = x, foo = anno))

# Add legends for `foo`
ht = Heatmap(m, top_annotation = HeatmapAnnotation(bar = x, foo = anno))
lgd = Legend(title = "foo", at = names(graphics), graphics = graphics)
draw(ht, annotation_legend_list = list(lgd))
```

`anno_density` *Density Annotation*

Description

Density Annotation

Usage

```
anno_density(x, which = c("column", "row"),
    type = c("lines", "violin", "heatmap"), xlim = NULL,
    heatmap_colors = rev(brewer.pal(name = "RdYlBu", n = 11)),
    joyplot_scale = 1, border = TRUE, gp = gpar(fill = "#CCCCCC"),
    axis = TRUE, axis_param = default_axis_param(which),
    width = NULL, height = NULL)
```

Arguments

`x`	A matrix or a list. If `x` is a matrix and if `which` is `column`, statistics for boxplots are calculated by columns, if `which` is `row`, the calculation is done by rows.
`which`	Whether it is a column annotation or a row annotation?
`type`	Type of graphics to represent density distribution. "lines" for normal density plot; "violine" for violin plot and "heatmap" for heatmap visualization of density distribution.
`xlim`	Range on x-axis.
`heatmap_colors`	A vector of colors for interpolating density values.
`joyplot_scale`	Relative height of density distribution. A value higher than 1 increases the height of the density distribution and the plot will represented as so-called "joyplot".
`border`	Wether draw borders of the annotation region?
`gp`	Graphic parameters for the boxes. The length of the graphic parameters should be one or the number of observations.
`axis`	Whether to add axis?
`axis_param`	parameters for controlling axis. See `default_axis_param` for all possible settings and default parameters.
`width`	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
`height`	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.

Value

An annotation function which can be used in HeatmapAnnotation.

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#density-annotation

Examples

```
m = matrix(rnorm(100), 10)
anno = anno_density(m, which = "row")
draw(anno, test = "normal density")
anno = anno_density(m, which = "row", type = "violin")
draw(anno, test = "violin")
anno = anno_density(m, which = "row", type = "heatmap")
draw(anno, test = "heatmap")
anno = anno_density(m, which = "row", type = "heatmap",
    heatmap_colors = c("white", "orange"))
draw(anno, test = "heatmap, colors")
```

---

anno_empty *Empty Annotation*

Description

Empty Annotation

Usage

```
anno_empty(which = c("column", "row"), border = TRUE, zoom = FALSE,
    width = NULL, height = NULL)
```

Arguments

`which`	Whether it is a column annotation or a row annotation?
`border`	Whether draw borders of the annotation region?
`zoom`	If it is true and when the heatmap is split, the empty annotation slices will have equal height or width, and you can see the correspondance between the annotation slices and the original heatmap slices.
`width`	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
`height`	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.

Details

It creates an empty annotation and holds space, later users can add graphics by decorate_annotation. This function is useful when users have difficulty to implement AnnotationFunction object.

In following example, an empty annotation is first created and later points are added:

```
m = matrix(rnorm(100), 10)
ht = Heatmap(m, top_annotation = HeatmapAnnotation(pt = anno_empty()))
ht = draw(ht)
co = column_order(ht)[[1]]
pt_value = 1:10
decorate_annotation("pt", {
pushViewport(viewport(xscale = c(0.5, ncol(mat)+0.5), yscale = range(pt_value)))
grid.points(seq_len(ncol(mat)), pt_value[co], pch = 16, default.units = "native")
grid.yaxis()
popViewport()
})
```

And it is similar as using anno_points:

```
Heatmap(m, top_annotation = HeatmapAnnotation(pt = anno_points(pt_value)))
```

Value

An annotation function which can be used in HeatmapAnnotation.

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#empty-annotation

Examples

```
anno = anno_empty()
draw(anno, test = "anno_empty")
anno = anno_empty(border = FALSE)
draw(anno, test = "anno_empty without border")
```

---

anno_histogram *Histogram Annotation*

Description

Histogram Annotation

Usage

```
anno_histogram(x, which = c("column", "row"), n_breaks = 11,
    border = FALSE, gp = gpar(fill = "#CCCCCC"),
    axis = TRUE, axis_param = default_axis_param(which),
    width = NULL, height = NULL)
```

Arguments

`x`	A matrix or a list. If `x` is a matrix and if `which` is `column`, statistics for boxplots are calculated by columns, if `which` is `row`, the calculation is done by rows.
`which`	Whether it is a column annotation or a row annotation?
`n_breaks`	Number of breaks for calculating histogram.
`border`	Wether draw borders of the annotation region?
`gp`	Graphic parameters for the boxes. The length of the graphic parameters should be one or the number of observations.
`axis`	Whether to add axis?
`axis_param`	parameters for controlling axis. See `default_axis_param` for all possible settings and default parameters.
`width`	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
`height`	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.

Value

An annotation function which can be used in `HeatmapAnnotation`.

See Also

`https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#histogram-annotation`

Examples

```
m = matrix(rnorm(1000), nc = 10)
anno = anno_histogram(t(m), which = "row")
draw(anno, test = "row histogram")
anno = anno_histogram(t(m), which = "row", gp = gpar(fill = 1:10))
draw(anno, test = "row histogram with color")
anno = anno_histogram(t(m), which = "row", n_breaks = 20)
draw(anno, test = "row histogram with color")
```

anno_horizon *Horizon chart Annotation*

Description

Horizon chart Annotation

Usage

```
anno_horizon(x, which = c("column", "row"),
    gp = gpar(pos_fill = "#D73027", neg_fill = "#313695"),
    n_slice = 4, slice_size = NULL, negative_from_top = FALSE,
    normalize = TRUE, gap = unit(0, "mm"),
    axis = TRUE, axis_param = default_axis_param(which),
    width = NULL, height = NULL)
```

Arguments

`x`	A matrix or a list. If `x` is a matrix or a data frame, columns correspond to observations.
`which`	Whether it is a column annotation or a row annotation?
`gp`	Graphic parameters for the boxes. The length of the graphic parameters should be one or the number of observations. There are two unstandard parameters specificly for horizon chart: `pos_fill` and `neg_fill` controls the filled color for positive values and negative values.
`n_slice`	Number of slices on y-axis.
`slice_size`	Height of the slice. If the value is not `NULL`, `n_slice` will be recalculated.
`negative_from_top`	Whether the areas for negative values start from the top or the bottom of the plotting region?
`normalize`	Whether normalize `x` by max(abs(x)).
`gap`	Gap size of neighbouring horizon chart.
`axis`	Whether to add axis?
`axis_param`	parameters for controlling axis. See `default_axis_param` for all possible settings and default parameters.
`width`	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
`height`	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.

Details

Horizon chart as row annotation is only supported.

Value

An annotation function which can be used in `HeatmapAnnotation`.

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#horizon-chart-annotation

Examples

```
lt = lapply(1:20, function(x) cumprod(1 + runif(1000, -x/100, x/100)) - 1)
anno = anno_horizon(lt, which = "row")
draw(anno, test = "horizon chart")
anno = anno_horizon(lt, which = "row",
    gp = gpar(pos_fill = "orange", neg_fill = "darkgreen"))
draw(anno, test = "horizon chart, col")
anno = anno_horizon(lt, which = "row", negative_from_top = TRUE)
draw(anno, test = "horizon chart + negative_from_top")
anno = anno_horizon(lt, which = "row", gap = unit(1, "mm"))
draw(anno, test = "horizon chart + gap")
anno = anno_horizon(lt, which = "row",
    gp = gpar(pos_fill = rep(c("orange", "red"), each = 10),
    neg_fill = rep(c("darkgreen", "blue"), each = 10)))
draw(anno, test = "horizon chart, col")
```

`anno_image` *Image Annotation*

Description

Image Annotation

Usage

```
anno_image(image, which = c("column", "row"), border = TRUE,
    gp = gpar(fill = NA, col = NA), space = unit(1, "mm"),
    width = NULL, height = NULL)
```

Arguments

`image`	A vector of file paths of images. The format of the image is inferred from the suffix name of the image file. NA values or empty strings in the vector means no image to drawn.
`which`	Whether it is a column annotation or a row annotation?
`border`	Wether draw borders of the annotation region?
`gp`	Graphic parameters for annotation grids. If the image has transparent background, the `fill` parameter can be used to control the background color in the annotation grids.

`space`	The space around the image to the annotation grid borders. The value should be a `unit` object.
`width`	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
`height`	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.

Details

This function supports image formats in `png`, `svg`, `pdf`, `eps`, `jpeg/jpg`, `tiff`. `png`, `jpeg/jpg` and `tiff` images are imported by `readPNG`, `readJPEG` and `readTIFF`, and drawn by `grid.raster`. `svg` images are firstly reformatted by `rsvg::rsvg_svg` and then imported by `readPicture` and drawn by `grid.picture`. `pdf` and `eps` images are imported by `PostScriptTrace` and `readPicture`, later drawn by `grid.picture`.

Different image formats can be mixed in the `image` vector.

Value

An annotation function which can be used in `HeatmapAnnotation`.

See Also

`https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#`
`image-annotation`

Examples

```
# download the free icons from https://github.com/Keyamoon/IcoMoon-Free
## Not run:
image = sample(dir("~/Downloads/IcoMoon-Free-master/PNG/64px", full.names = TRUE), 10)
anno = anno_image(image)
draw(anno, test = "png")
image[1:5] = ""
anno = anno_image(image)
draw(anno, test = "some of png")

## End(Not run)
```

`anno_joyplot` *Joyplot Annotation*

Description

Joyplot Annotation

Usage

```
anno_joyplot(x, which = c("column", "row"), gp = gpar(fill = "#000000"),
    scale = 2, transparency = 0.6,
    axis = TRUE, axis_param = default_axis_param(which),
    width = NULL, height = NULL)
```

Arguments

x	A matrix or a list. If x is a matrix or a data frame, columns correspond to observations.
which	Whether it is a column annotation or a row annotation?
gp	Graphic parameters for the boxes. The length of the graphic parameters should be one or the number of observations.
scale	Relative height of the curve. A value higher than 1 increases the height of the curve.
transparency	Transparency of the filled colors. Value should be between 0 and 1.
axis	Whether to add axis?
axis_param	parameters for controlling axis. See default_axis_param for all possible settings and default parameters.
width	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
height	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.

Value

An annotation function which can be used in HeatmapAnnotation.

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#joyplot-annotation

Examples

```
m = matrix(rnorm(1000), nc = 10)
lt = apply(m, 2, function(x) data.frame(density(x)[c("x", "y")]))
anno = anno_joyplot(lt, width = unit(4, "cm"), which = "row")
draw(anno, test = "joyplot")
anno = anno_joyplot(lt, width = unit(4, "cm"), which = "row", gp = gpar(fill = 1:10))
draw(anno, test = "joyplot + col")
anno = anno_joyplot(lt, width = unit(4, "cm"), which = "row", scale = 1)
draw(anno, test = "joyplot + scale")

m = matrix(rnorm(5000), nc = 50)
lt = apply(m, 2, function(x) data.frame(density(x)[c("x", "y")]))
anno = anno_joyplot(lt, width = unit(4, "cm"), which = "row", gp = gpar(fill = NA), scale = 4)
draw(anno, test = "joyplot")
```

anno_lines *Lines Annotation*

Description

Lines Annotation

Usage

```
anno_lines(x, which = c("column", "row"), border = TRUE, gp = gpar(),
   add_points = smooth, smooth = FALSE, pch = 16, size = unit(2, "mm"), pt_gp = gpar(), ylim = NULL,
     extend = 0.05, axis = TRUE, axis_param = default_axis_param(which),
     width = NULL, height = NULL)
```

Arguments

`x`	The value vector. The value can be a vector or a matrix. The length of the vector or the number of rows of the matrix is taken as the number of the observations of the annotation.
`which`	Whether it is a column annotation or a row annotation?
`border`	Wether draw borders of the annotation region?
`gp`	Graphic parameters for lines. The length of each graphic parameter can be 1, or number of columns of `x` is `x` is a matrix.
`add_points`	Whether to add points on the lines?
`smooth`	If it is `TRUE`, smoothing by `loess` is performed. If it is `TRUE`, `add_points` is set to `TRUE` by default.
`pch`	Point type. The length setting is the same as `gp`.
`size`	Point size, the value should be a `unit` object. The length setting is the same as `gp`.
`pt_gp`	Graphic parameters for points. The length setting is the same as `gp`.
`ylim`	Data ranges. By default it is `range(x)`.
`extend`	The extension to both side of `ylim`. The value is a percent value corresponding to `ylim[2] -ylim[1]`.
`axis`	Whether to add axis?
`axis_param`	parameters for controlling axis. See `default_axis_param` for all possible settings and default parameters.
`width`	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
`height`	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.

Value

An annotation function which can be used in `HeatmapAnnotation`.

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#lines-annotation

Examples

```
anno = anno_lines(runif(10))
draw(anno, test = "anno_lines")
anno = anno_lines(cbind(c(1:5, 1:5), c(5:1, 5:1)), gp = gpar(col = 2:3))
draw(anno, test = "matrix")
anno = anno_lines(cbind(c(1:5, 1:5), c(5:1, 5:1)), gp = gpar(col = 2:3),
add_points = TRUE, pt_gp = gpar(col = 5:6), pch = c(1, 16))
draw(anno, test = "matrix")
```

---

anno_link *Link Annotation*

Description

Link Annotation

Usage

```
anno_link(...)
```

Arguments

...	Pass to anno_zoom.

Details

This function is the same as anno_zoom. It links subsets of rows or columns to a list of graphic regions.

Examples

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

anno_mark — *Link annotation with labels*

Description

Link annotation with labels

Usage

```
anno_mark(at, labels, which = c("column", "row"),
    side = ifelse(which == "column", "top", "right"),
    lines_gp = gpar(), labels_gp = gpar(),
    labels_rot = ifelse(which == "column", 90, 0), padding = unit(1, "mm"),
    link_width = unit(5, "mm"), link_height = link_width,
    link_gp = lines_gp,
    extend = unit(0, "mm"))
```

Arguments

`at`	Numeric index from the original matrix.
`labels`	Corresponding labels.
`which`	Whether it is a column annotation or a row annotation?
`side`	Side of the labels. If it is a column annotation, valid values are "top" and "bottom"; If it is a row annotation, valid values are "left" and "right".
`lines_gp`	Please use `link_gp` instead.
`link_gp`	Graphic settings for the segments.
`labels_gp`	Graphic settings for the labels.
`labels_rot`	Rotations of labels, scalar.
`padding`	Padding between neighbouring labels in the plot.
`link_width`	Width of the segments.
`link_height`	Similar as `link_width`, used for column annotation.
`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 `unit` 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

An annotation function which can be used in `HeatmapAnnotation`.

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#mark-annotation

Examples

```
anno = anno_mark(at = c(1:4, 20, 60, 97:100), labels = month.name[1:10], which = "row")
draw(anno, index = 1:100, test = "anno_mark")

m = matrix(1:1000, byrow = TRUE, nr = 100)
anno = anno_mark(at = c(1:4, 20, 60, 97:100), labels = month.name[1:10], which = "row")
Heatmap(m, cluster_rows = FALSE, cluster_columns = FALSE) + rowAnnotation(mark = anno)
Heatmap(m) + rowAnnotation(mark = anno)
```

`anno_oncoprint_barplot` *Barplot Annotation for oncoPrint*

Description

Barplot Annotation for oncoPrint

Usage

```
anno_oncoprint_barplot(type = NULL, which = c("column", "row"),
   bar_width = 0.6, beside = FALSE, ylim = NULL, show_fraction = FALSE, axis = TRUE,
   axis_param = if(which == "column") default_axis_param("column") else list(side = "top", labels_rot =
    width = NULL, height = NULL, border = FALSE)
```

Arguments

type	A vector of the alteration types in the data. It can be a subset of all alteration types if you don't want to show them all.
which	Is it a row annotation or a column annotation?
bar_width	Width of the bars.
beside	Will bars be stacked or be positioned beside each other?
ylim	Data range.
show_fraction	Whether to show the numbers or the fractions?
axis	Whether draw axis?
axis_param	Parameters for controlling axis.
width	Width of the annotation.
height	Height of the annotation.
border	Whether draw the border?

Details

This annotation function should always be used with `oncoPrint`.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

`anno_points` *Points Annotation*

Description

Points Annotation

Usage

```
anno_points(x, which = c("column", "row"), border = TRUE, gp = gpar(), pch = 16,
    size = unit(2, "mm"), ylim = NULL, extend = 0.05, axis = TRUE,
    axis_param = default_axis_param(which), width = NULL, height = NULL, ...)
```

Arguments

`x`	The value vector. The value can be a vector or a matrix. The length of the vector or the number of rows of the matrix is taken as the number of the observations of the annotation.
`which`	Whether it is a column annotation or a row annotation?
`border`	Wether draw borders of the annotation region?
`gp`	Graphic parameters for points. The length of each graphic parameter can be 1, length of `x` if `x` is a vector, or number of columns of `x` is `x` is a matrix.
`pch`	Point type. The length setting is the same as `gp`.
`size`	Point size, the value should be a `unit` object. The length setting is the same as `gp`.
`ylim`	Data ranges. By default it is `range(x)`.
`extend`	The extension to both side of `ylim`. The value is a percent value corresponding to `ylim[2] -ylim[1]`.
`axis`	Whether to add axis?
`axis_param`	parameters for controlling axis. See `default_axis_param` for all possible settings and default parameters.

`width`	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
`height`	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.
`...`	Other arguments.

Value

An annotation function which can be used in `HeatmapAnnotation`.

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#points-annotation

Examples

```
anno = anno_points(runif(10))
draw(anno, test = "anno_points")
anno = anno_points(matrix(runif(20), nc = 2), pch = 1:2)
draw(anno, test = "matrix")
```

`anno_simple` *Simple Annotation*

Description

Simple Annotation

Usage

```
anno_simple(x, col, na_col = "grey",
    which = c("column", "row"), border = FALSE, gp = gpar(),
    pch = NULL, pt_size = unit(1, "snpc")*0.8, pt_gp = gpar(),
    simple_anno_size = ht_opt$simple_anno_size,
    width = NULL, height = NULL)
```

Arguments

`x`	The value vector. The value can be a vector or a matrix. The length of the vector or the nrow of the matrix is taken as the number of the observations of the annotation. The value can be numeric or character and NA value is allowed.
`col`	Color that maps to `x`. If `x` is numeric and needs a continuous mapping, `col` should be a color mapping function which accepts a vector of values and returns a vector of colors. Normally it is generated by `colorRamp2`. If `x` is discrete (numeric or character) and needs a discrete color mapping, `col` should be a vector of colors with levels in `x` as vector names. If `col` is not specified, the color mapping is randomly generated by `ComplexHeatmap:::default_col`.

`na_col`	Color for NA value.
`which`	Whether it is a column annotation or a row annotation?
`border`	Wether draw borders of the annotation region?
`gp`	Graphic parameters for grid borders. The `fill` parameter is disabled.
`pch`	Points/symbols that are added on top of the annotation grids. The value can be numeric or single letters. It can be a vector if `x` is a vector and a matrix if `x` is a matrix. No points are drawn if the corresponding values are NA.
`pt_size`	Size of the points/symbols. It should be a `unit` object. If `x` is a vector, the value of `pt_size` can be a vector, while if `x` is a matrix, `pt_size` can only be a single value.
`pt_gp`	Graphic parameters for points/symbols. The length setting is same as `pt_size`. If `pch` is set as letters, the fontsize should be set as `pt_gp = gpar(fontsize = ...)`.
`simple_anno_size`	size of the simple annotation.
`width`	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
`height`	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.

Details

The "simple annotation" is the most widely used annotation type which is heatmap-like, where the grid colors correspond to the values. `anno_simple` also supports to add points/symbols on top of the grids where the it can be normal point (when pch is set as numbers) or letters (when pch is set as single letters).

Value

An annotation function which can be used in `HeatmapAnnotation`.

See Also

`https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#simple-annotation-as-an-annotation-function`

Examples

```
anno = anno_simple(1:10)
draw(anno, test = "a numeric vector")

anno = anno_simple(cbind(1:10, 10:1))
draw(anno, test = "a matrix")

anno = anno_simple(1:10, pch = c(1:4, NA, 6:8, NA, 10))
draw(anno, test = "pch has NA values")

anno = anno_simple(1:10, pch = c(rep("A", 5), rep(NA, 5)))
```

```
draw(anno, test = "pch has NA values")

pch = matrix(1:20, nc = 2)
pch[sample(length(pch), 10)] = NA
anno = anno_simple(cbind(1:10, 10:1), pch = pch)
draw(anno, test = "matrix, pch is a matrix with NA values")
```

---

anno_summary *Summary Annotation*

Description

Summary Annotation

Usage

```
anno_summary(which = c("column", "row"), border = TRUE, bar_width = 0.8,
    axis = TRUE, axis_param = default_axis_param(which),
    ylim = NULL, extend = 0.05, outline = TRUE, box_width = 0.6,
    pch = 1, size = unit(2, "mm"), gp = gpar(),
    width = NULL, height = NULL)
```

Arguments

`which`	Whether it is a column annotation or a row annotation?
`border`	Wether draw borders of the annotation region?
`bar_width`	Relative width of the bars. The value should be smaller than one.
`axis`	Whether to add axis?
`axis_param`	parameters for controlling axis. See `default_axis_param` for all possible settings and default parameters.
`ylim`	Data ranges. `ylim` for barplot is enforced to be `c(0,1)`.
`extend`	The extension to both side of `ylim`. The value is a percent value corresponding to `ylim[2] -ylim[1]`. This argument is only for boxplot.
`outline`	Whether draw outline of boxplots?
`box_width`	Relative width of boxes. The value should be smaller than one.
`pch`	Point style.
`size`	Point size.
`gp`	Graphic parameters.
`width`	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
`height`	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.

Details

`anno_summary` is a special annotation function that it only works for one-column or one-row heatmap. It shows the summary of the values in the heatmap. If the values in the heatmap is discrete, the proportion of each level (the sum is normalized to 1) is visualized as stacked barplot. If the heatmap is split into multiple slices, multiple bars are put in the annotation. If the value is continuous, boxplot is used.

In the barplot, the color schema is used as the same as the heatmap, while for the boxplot, the color needs to be controlled by gp.

Value

An annotation function which can be used in `HeatmapAnnotation`.

See Also

`https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#summary-annotation`

Examples

```
ha = HeatmapAnnotation(summary = anno_summary(height = unit(4, "cm")))
v = sample(letters[1:2], 50, replace = TRUE)
split = sample(letters[1:2], 50, replace = TRUE)
Heatmap(v, top_annotation = ha, width = unit(1, "cm"), split = split)

ha = HeatmapAnnotation(summary = anno_summary(gp = gpar(fill = 2:3), height = unit(4, "cm")))
v = rnorm(50)
Heatmap(v, top_annotation = ha, width = unit(1, "cm"), split = split)
```

`anno_text` *Text Annotation*

Description

Text Annotation

Usage

```
anno_text(x, which = c("column", "row"), gp = gpar(),
    rot = guess_rot(), just = guess_just(),
    offset = guess_location(), location = guess_location(),
    width = NULL, height = NULL, show_name = FALSE)
```

Arguments

x	A vector of text.
which	Whether it is a column annotation or a row annotation?
gp	Graphic parameters.
rot	Rotation of the text, pass to grid.text.
just	Justification of text, pass to grid.text.
offset	Depracated, use location instead.
location	Position of the text. By default rot, just and location are automatically inferred according to whether it is a row annotation or column annotation. The value of location should be a unit object, normally in npc unit. E.g. unit(0,'npc') means the most left of the annotation region and unit(1,'npc') means the most right of the annotation region.
width	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.
height	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.
show_name	Whether to show the annotation name.

Value

An annotation function which can be used in HeatmapAnnotation.

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#text-annotation

Examples

```
anno = anno_text(month.name)
draw(anno, test = "month names")
anno = anno_text(month.name, gp = gpar(fontsize = 16))
draw(anno, test = "month names with fontsize")
anno = anno_text(month.name, gp = gpar(fontsize = 1:12+4))
draw(anno, test = "month names with changing fontsize")
anno = anno_text(month.name, which = "row")
draw(anno, test = "month names on rows")
anno = anno_text(month.name, location = 0, rot = 45,
    just = "left", gp = gpar(col = 1:12))
draw(anno, test = "with rotations")
anno = anno_text(month.name, location = 1,
    rot = 45, just = "right", gp = gpar(fontsize = 1:12+4))
draw(anno, test = "with rotations")
```

anno_zoom *Zoom annotation*

Description

Zoom annotation

Usage

```
anno_zoom(align_to, panel_fun = function(index, nm = NULL) { grid.rect() },
    which = c("column", "row"), side = ifelse(which == "column", "top", "right"),
     size = NULL, gap = unit(1, "mm"),
     link_width = unit(5, "mm"), link_height = link_width, link_gp = gpar(),
     extend = unit(0, "mm"), width = NULL, height = NULL, internal_line = TRUE)
```

Arguments

`align_to`	It defines how the boxes correspond to the rows or the columns in the heatmap. If the value is a list of indices, each box corresponds to the rows or columns with indices in one vector in the list. If the value is a categorical variable (e.g. a factor or a character vector) that has the same length as the rows or columns in the heatmap, each box corresponds to the rows/columns in each level in the categorical variable.
`panel_fun`	A self-defined function that defines how to draw graphics in the box. The function must have a `index` argument which is the indices for the rows/columns that the box corresponds to. It can have second argument `nm` which is the "name" of the selected part in the heatmap. The corresponding value for `nm` comes from `align_to` if it is specified as a categorical variable or a list with names.
`which`	Whether it is a column annotation or a row annotation?
`side`	Side of the boxes If it is a column annotation, valid values are "top" and "bottom"; If it is a row annotation, valid values are "left" and "right".
`size`	The size of boxes. It can be pure numeric that they are treated as relative fractions of the total height/width of the heatmap. The value of `size` can also be absolute units.
`gap`	Gaps between boxes.
`link_gp`	Graphic settings for the segments.
`link_width`	Width of the segments.
`link_height`	Similar as `link_width`, used for column annotation.
`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 `unit` object. The length can be either one or two.
`width`	Width of the annotation. The value should be an absolute unit. Width is not allowed to be set for column annotation.

height	Height of the annotation. The value should be an absolute unit. Height is not allowed to be set for row annotation.
internal_line	Internally used.

Details

anno_zoom creates several plotting regions (boxes) which can be corresponded to subsets of rows/columns in the heatmap.

Value

An annotation function which can be used in HeatmapAnnotation.

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html#zoom-annotation

Examples

```
set.seed(123)
m = matrix(rnorm(100*10), nrow = 100)
subgroup = sample(letters[1:3], 100, replace = TRUE, prob = c(1, 5, 10))
rg = range(m)
panel_fun = function(index, nm) {
pushViewport(viewport(xscale = rg, yscale = c(0, 2)))
grid.rect()
grid.xaxis(gp = gpar(fontsize = 8))
grid.boxplot(m[index, ], pos = 1, direction = "horizontal")
grid.text(paste("distribution of group", nm), mean(rg), y = 1.9,
just = "top", default.units = "native", gp = gpar(fontsize = 10))
popViewport()
}
anno = anno_zoom(align_to = subgroup, which = "row", panel_fun = panel_fun,
size = unit(2, "cm"), gap = unit(1, "cm"), width = unit(4, "cm"))
Heatmap(m, right_annotation = rowAnnotation(foo = anno), row_split = subgroup)
```

---

attach_annotation-Heatmap-method

Attach heatmap annotations to the heatmap

Description

Attach heatmap annotations to the heatmap

Usage

```
## S4 method for signature 'Heatmap'
attach_annotation(object, ha, side = c("top", "bottom", "left", "right"),
    gap = unit(1, "points"))
```

Arguments

`object`	A `Heatmap-class` object.
`ha`	A `HeatmapAnnotation-class` object.
`side`	Which side of the heatmap. Value should be in "top", "bottom", "left", "right".
`gap`	Space between the two heatmap annotations.

Examples

```
m = matrix(rnorm(100), 10)
ht = Heatmap(m)
ha = HeatmapAnnotation(foo = 1:10)
ht = attach_annotation(ht, ha)
ht
ha2 = HeatmapAnnotation(bar = letters[1:10])
ht = attach_annotation(ht, ha2)
ht
```

`bar3D` *Draw 3D bars*

Description

Draw 3D bars

Usage

```
bar3D(x, y, w, h, l, theta = 60, default.units = "npc", fill = "white", col = "black")
```

Arguments

`x`	x coordinate of the center point in the bottom face.
`y`	y coordinate of the center point in the bottom face.
`w`	Width of the botton face.
`h`	Height of the botton face.
`l`	Length of the bars (in the z-direction).
`theta`	The angle for the projection.
`default.units`	Units.
`fill`	Filled colors for the bars.
`col`	Border colors.

Examples

```
grid.newpage()
bar3D(c(0.3, 0.7), 0.5, 0.2, 0.2, 0.2, fill = 2:3)
```

bin_genome — *Bin the genome*

Description

Bin the genome

Usage

```
bin_genome(species = "hg19", bins = 2000, bin_size = NULL, ...)
```

Arguments

species	Abbreviation of the genome, pass to read.chromInfo.
bins	Number of bins. The final number of bins is approximately equal to it.
bin_size	Size of the bins. If bin_size is set, bins is ignored.
...	All pass to read.chromInfo. E.g. you can set a subset of chromosomes there.

Value

A GRanges object of the genomic bins.

Examples

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

c.ColorMapping — *Concatenate A List of ColorMapping objects*

Description

Concatenate A List of ColorMapping objects

Usage

```
## S3 method for class 'ColorMapping'
c(..., name = NULL)
```

Arguments

...	A list of ColorMapping-class objects.
name	Name of the new merged color mapping.

Details

Only discrete color mappings can be concatenated.

Examples

```
cm1 = ColorMapping(colors = c("A" = "red", "B" = "black"))
cm2 = ColorMapping(colors = c("B" = "blue", "C" = "green"))
c(cm1, cm2)
```

`c.HeatmapAnnotation` *Concatenate Heatmap Annotations*

Description

Concatenate Heatmap Annotations

Usage

```
## S3 method for class 'HeatmapAnnotation'
c(..., gap = unit(1, "points"))
```

Arguments

`...`	`HeatmapAnnotation-class` objects.
`gap`	Gap between the groups of annotations.

Details

The heatmap annotations should have same number of observations.

Examples

```
ha1 = HeatmapAnnotation(foo = 1:10)
ha2 = HeatmapAnnotation(bar = anno_points(10:1))
ha = c(ha1, ha2)
ha
ha3 = HeatmapAnnotation(sth = cbind(1:10, 10:1))
ha = c(ha1, ha2, ha3, gap = unit(c(1, 4), "mm"))
ha
```

cluster_between_groups *Cluster only between Groups*

Description

Cluster only between Groups

Usage

```
cluster_between_groups(mat, factor)
```

Arguments

`mat`	A matrix where clustering is applied on columns.
`factor`	A categorical vector.

Details

The clustering is only applied between groups and inside a group, the order is unchanged.

Value

A `dendrogram` object.

Examples

```
m = matrix(rnorm(120), nc = 12)
colnames(m) = letters[1:12]
fa = rep(c("a", "b", "c"), times = c(2, 4, 6))
dend = cluster_between_groups(m, fa)
grid.dendrogram(dend, test = TRUE)
```

cluster_within_group *Cluster within and between Groups*

Description

Cluster within and between Groups

Usage

```
cluster_within_group(mat, factor)
```

Arguments

`mat`	A matrix where clustering is applied on columns.
`factor`	A categorical vector.

Details

The clustering is firstly applied in each group, then clustering is applied to group means. The within-group dendrograms and between-group dendrogram are finally connected by `merge_dendrogram`.

In the final dendrogram, the within group dendrograms are enforced to be flat lines to emphasize that the within group dendrograms have no sense to compare to between-group dendrogram.

Value

A `dendrogram` object. The order of columns can be retrieved by `order.dendrogram`.

Examples

```
m = matrix(rnorm(120), nc = 12)
colnames(m) = letters[1:12]
fa = rep(c("a", "b", "c"), times = c(2, 4, 6))
dend = cluster_within_group(m, fa)
grid.dendrogram(dend, test = TRUE)
```

`ColorMapping` *Constructor Method for ColorMapping Class*

Description

Constructor Method for ColorMapping Class

Usage

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

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 `colorRamp2`, `breaks` is automatically inferred from the color mapping function.
`na_col`	Colors for `NA` values.
`full_col`	A super set of `colors`, used internally.

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

```
cm = ColorMapping(colors = c("A" = "red", "B" = "black"))
cm
require(circlize)
col_fun = colorRamp2(c(0, 1), c("white", "red"))
cm = ColorMapping(col_fun = col_fun)
```

`ColorMapping-class` *Class for Color Mapping*

Description

Class for Color Mapping

Details

The `ColorMapping-class` handles color mapping for 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`: contructor methods.
- `map_to_colors,ColorMapping-method`: mapping values to colors.
- `color_mapping_legend,ColorMapping-method`: draw legend or get legend as an object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
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, ...,

    color_bar = object@type,

    title = object@name,
    title_gp = gpar(fontsize = 10, fontface = "bold"),
    title_position = "topleft",
    grid_height = unit(4, "mm"),
    grid_width = unit(4, "mm"),
    border = NULL,
    at = object@levels,
    labels = at,
    labels_gp = gpar(fontsize = 10),
    labels_rot = 0,
    nrow = NULL,
    ncol = 1,
    by_row = FALSE,
    legend_height = NULL,
    legend_width = NULL,
    legend_direction = c("vertical", "horizontal"),
    break_dist = NULL,

    graphics = NULL,
    param = NULL)
```

Arguments

`object`	A `ColorMapping-class` object.
`plot`	Whether to plot or just return the legend object?
`...`	Pass to `draw,Legends-method`.
`color_bar`	"continous" or "discrete". It controls whether to show the discrete legend for the continuous color mapping.
`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. See `Legend` for all possible values.
`grid_height`	Height of each legend grid. Pass to `Legend`.
`grid_width`	Width of each legend grid. Pass to `Legend`.
`border`	Color for legend grid borders. Pass to `Legend`.
`at`	Break values of the legend. By default it is the levels in the `ColorMapping-class` object.
`labels`	Labels corresponding to break values.
`labels_gp`	Graphcial parameters for legend labels.
`labels_rot`	Rotation of labels.
`nrow`	Pass to `Legend`. It controls the layout of legend grids if they are arranged in multiple rows or columns.
`ncol`	Pass to `Legend`. It controls the layout of legend grids if they are arranged in multiple rows or columns.
`by_row`	Pass to `Legend`. It controls the order of legend grids if they are arranged in multiple rows or columns.
`legend_height`	Height of the legend body. It only works when `color_bar` is `continuous` and `direction` is `vertical`. Pass to `Legend`.
`legend_width`	Width of the legend body. It only works when `color_bar` is `continuous` and `direction` is `horizontal`. Pass to `Legend`.
`legend_direction`	When `color_bar` is `continuous`, whether the legend is vertical or horizontal? Pass to `Legend`.
`break_dist`	A zooming factor to control relative distance of two neighbouring break values.The length of it should be `length(at) -1` or a scalar.
`graphics`	Internally used.
`param`	All the legend-related parameters can be specified as a single list.

Details

The legend is constructed by `Legend`.

Value

A `Legends-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

`columnAnnotation` *Construct Column Annotations*

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

```
# 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,Heatmap-method`, `Heatmap-class` class method
- `column_dend,HeatmapList-method`, `HeatmapList-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

The format of the returned object depends on whether rows/columns of the heatmaps are split.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

column_dend-HeatmapList-method

Get Column Dendrograms from a hHeatmap List

Description

Get Column Dendrograms from a hHeatmap List

Usage

```
## S4 method for signature 'HeatmapList'
column_dend(object, name = NULL)
```

Arguments

`object`	A `HeatmapList-class` object.
`name`	Name of a specific heatmap.

Value

The format of the returned object depends on whether rows/columns of the heatmaps are split.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
mat = matrix(rnorm(100), 10)
ht_list = Heatmap(mat) + Heatmap(mat)
ht_list = draw(ht_list)
column_dend(ht_list)
ht_list = Heatmap(mat, column_km = 2) + Heatmap(mat, column_km = 2)
ht_list = draw(ht_list)
column_dend(ht_list)
ht_list = Heatmap(mat) %v% Heatmap(mat)
ht_list = draw(ht_list)
column_dend(ht_list)
ht_list = Heatmap(mat, column_km = 2) %v% Heatmap(mat)
ht_list = draw(ht_list)
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,Heatmap-method`, `Heatmap-class` class method
- `column_order,HeatmapList-method`, `HeatmapList-class` class method

Examples

```
# no example
NULL
```

`column_order-Heatmap-method` *Get Column Order from a Aeatmap List*

Description

Get Column Order from a Aeatmap List

Usage

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

Arguments

`object`	A `Heatmap-class` object.

Value

The format of the returned object depends on whether rows/columns of the heatmaps are split.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
mat = matrix(rnorm(100), 10)
ht = Heatmap(mat)
ht = draw(ht)
column_order(ht)
ht = Heatmap(mat, column_km = 2)
ht = draw(ht)
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, name = NULL)
```

Arguments

`object`	A `HeatmapList-class` object.
`name`	Name of a specific heatmap.

Value

The format of the returned object depends on whether rows/columns of the heatmaps are split.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
mat = matrix(rnorm(100), 10)
ht_list = Heatmap(mat) + Heatmap(mat)
ht_list = draw(ht_list)
column_order(ht_list)
ht_list = Heatmap(mat, column_km = 2) + Heatmap(mat, column_km = 2)
ht_list = draw(ht_list)
column_order(ht_list)
ht_list = Heatmap(mat) %v% Heatmap(mat)
ht_list = draw(ht_list)
column_order(ht_list)
ht_list = Heatmap(mat, column_km = 2) %v% Heatmap(mat)
ht_list = draw(ht_list)
column_order(ht_list)
```

comb_degree *Degrees of the Combination sets*

Description

Degrees of the Combination sets

Usage

```
comb_degree(m)
```

Arguments

m	A combination matrix returned by `make_comb_mat`.

Details

The degree for a combination set is the number of sets that are selected.

Value

A vector of degrees of the combination sets.

Examples

```
set.seed(123)
lt = list(a = sample(letters, 10),
          b = sample(letters, 15),
          c = sample(letters, 20))
m = make_comb_mat(lt)
comb_degree(m)
```

comb_name *Names of the Combination sets*

Description

Names of the Combination sets

Usage

```
comb_name(m, readable = FALSE)
```

Arguments

m	A combination matrix returned by `make_comb_mat`.
readable	Whether the combination represents as e.g. "A&B&C".

Details

The name of the combination sets are formatted as a string of binary bits. E.g. for three sets of "a", "b", "c", the combination set with name "101" corresponds to select set a, not select set b and select set c. The definition of "select" depends on the value of `mode` from `make_comb_mat`.

Value

A vector of names of the combination sets.

Examples

```
set.seed(123)
lt = list(a = sample(letters, 10),
          b = sample(letters, 15),
          c = sample(letters, 20))
m = make_comb_mat(lt)
comb_name(m)
comb_name(m, readable = TRUE)
```

`comb_size` *Sizes of the Combination sets*

Description

Sizes of the Combination sets

Usage

```
comb_size(m, degree = NULL)
```

Arguments

`m`	A combination matrix returned by `make_comb_mat`.
`degree`	degree of the intersection. The value can be a vector.

Value

A vector of sizes of the combination sets.

Examples

```
set.seed(123)
lt = list(a = sample(letters, 10),
          b = sample(letters, 15),
          c = sample(letters, 20))
m = make_comb_mat(lt)
comb_size(m)
```

compare_heatmap — *Compare heatmaps between stats::heatmap() and ComplexHeatmap::heatmap()*

Description

Compare heatmaps between stats::heatmap() and ComplexHeatmap::heatmap()

Usage

```
compare_heatmap(...)
```

Arguments

`...`	The same set of arguments passed to `stats::heatmap` and `ComplexHeatmap::heatmap`.

Details

The function plots two heatmaps, one by `stats::heatmap` and one by `ComplexHeatmap::heatmap`. Users can see the difference between the two implementations.

Examples

```
mat = matrix(rnorm(100), 10)
compare_heatmap(mat)
```

compare_heatmap.2 — *Compare heatmaps between gplots::heatmap.2() and ComplexHeatmap::heatmap()*

Description

Compare heatmaps between gplots::heatmap.2() and ComplexHeatmap::heatmap()

Usage

```
compare_heatmap.2(...)
```

Arguments

`...`	The same set of arguments passed to `gplots::heatmap.2` and `ComplexHeatmap::heatmap.2`.

Details

The function plots two heatmaps, one by `gplots::heatmap.2` and one by `ComplexHeatmap::heatmap.2`. Users can see the difference between the two implementations.

Examples

```
mat = matrix(rnorm(100), 10)
compare_heatmap.2(mat)
```

---

compare_pheatmap *Compare heatmaps between pheatmap::pheatmap() and ComplexHeatmap::pheatmap()*

Description

Compare heatmaps between pheatmap::pheatmap() and ComplexHeatmap::pheatmap()

Usage

```
compare_pheatmap(...)
```

Arguments

`...`	The same set of arguments passed to `pheatmap::pheatmap` and `ComplexHeatmap::pheatmap`.

Details

The function plots two heatmaps, one by `pheatmap::pheatmap` and one by `ComplexHeatmap::pheatmap`. Users can see the difference between the two implementations.

Examples

```
mat = matrix(rnorm(100), 10)
compare_pheatmap(mat)
```

---

complement_size *Complement Set Size*

Description

Complement Set Size

Usage

```
complement_size(m)
```

Arguments

`m`	A combination matrix returned by `make_comb_mat`.

Value

If there is no complement set, it returns zero.

Examples

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

`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`

Heights of Heatmap Components

Description

Heights of Heatmap Components

Usage

```
## S4 method for signature 'Heatmap'
component_height(object, k = HEATMAP_LAYOUT_COLUMN_COMPONENT)
```

Arguments

`object`	A `Heatmap-class` object.
`k`	Which components in the heatmap. The value should numeric indices or the names of the corresponding column component. See **Detials**.

Details

All column components are: `column_title_top`, `column_dend_top`, `column_names_top`, `column_anno_top`, `heatmap_body`, `column_anno_bottom`, `column_names_bottom`, `column_dend_bottom`, `column_title_bottom`.

This function is only for internal use.

Value

A `unit` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

`component_height-HeatmapList-method`
Height of Heatmap List Components

Description

Height of Heatmap List Components

Usage

```
## S4 method for signature 'HeatmapList'
component_height(object, k = HEATMAP_LIST_LAYOUT_COLUMN_COMPONENT)
```

Arguments

`object`	A `HeatmapList-class` object.
`k`	Which component in the heatmap list. Values are in `ComplexHeatmap:::HEATMAP_LIST_LAYOUT_COLUMN`

Value

A `unit` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

---

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,Heatmap-method`, `Heatmap-class` class method
- `component_width,HeatmapList-method`, `HeatmapList-class` class method

Examples

```
# no example
NULL
```

---

component_width-Heatmap-method

Widths of Heatmap Components

Description

Widths of Heatmap Components

Usage

```
## S4 method for signature 'Heatmap'
component_width(object, k = HEATMAP_LAYOUT_ROW_COMPONENT)
```

Arguments

`object`	A `Heatmap-class` object.
`k`	Which components in the heatmap. The value should numeric indices or the names of the corresponding row component. See **Detials**.

Details

All row components are: `row_title_left`, `row_dend_left`, `row_names_left`, `row_anno_left`, `heatmap_body`, `row_anno_right`, `row_names_right`, `row_dend_right`, `row_title_right`.

This function is only for internal use.

Value

A `unit` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

`component_width-HeatmapList-method`
Width of Heatmap List Components

Description

Width of Heatmap List Components

Usage

```
## S4 method for signature 'HeatmapList'
component_width(object, k = HEATMAP_LIST_LAYOUT_ROW_COMPONENT)
```

Arguments

`object`	A `HeatmapList-class` object.
`k`	Which component in the heatmap list. Values are in `ComplexHeatmap:::HEATMAP_LIST_LAYOUT_ROW_CO`

Details

This function is only for internal use.

Value

A unit object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

---

copy_all-AnnotationFunction-method

Copy the AnnotationFunction Object

Description

Copy the AnnotationFunction Object

Usage

```
## S4 method for signature 'AnnotationFunction'
copy_all(object)
```

Arguments

object	The AnnotationFunction-class object.

Details

In AnnotationFunction-class, there is an environment which stores some external variables for the annotation function (specified by the `var_import` argument when constructing the AnnotationFunction-class object. This copy_all,AnnotationFunction-method hard copies all the variables into a new isolated environment.

The environment is at `object@var_env`.

Examples

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

`copy_all-dispatch` *Method dispatch page for copy_all*

Description

Method dispatch page for `copy_all`.

Dispatch

`copy_all` can be dispatched on following classes:

- `copy_all,AnnotationFunction-method`, `AnnotationFunction-class` class method
- `copy_all,SingleAnnotation-method`, `SingleAnnotation-class` class method

Examples

```
# no example
NULL
```

`copy_all-SingleAnnotation-method` *Copy the SingleAnnotation object*

Description

Copy the SingleAnnotation object

Usage

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

Arguments

`object`	The `SingleAnnotation-class` object.

Details

Since the SingleAnnotation object always contains an `AnnotationFunction-class` object, it calls `copy_all,AnnotationFunction-method` to hard copy the variable environment.

Examples

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

decorate_annotation *Decorate Heatmap Annotation*

Description

Decorate Heatmap Annotation

Usage

```
decorate_annotation(annotation, code, slice = 1, envir = new.env(parent = parent.frame()))
```

Arguments

annotation	Name of the annotation.
code	Code that adds graphics in the selected heatmap annotation.
slice	Index of the row slices or the column slice 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 contructs the name of the viewport, goes to the viewport by `seekViewport`, runs code to that viewport, and finally goes back to the original viewport.

Value

The function returns no value.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-decoration.html

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 Column Dendrograms*

Description

Decorate Heatmap Column Dendrograms

Usage

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

Arguments

`...`	Pass to `decorate_dend`.
`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

```
# There is no example
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 `decorate_dimnames`.
`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

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

`decorate_column_title` *Decorate Heatmap Column Titles*

Description

Decorate Heatmap Column Titles

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

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

`decorate_dend` *Decorate Heatmap Dendrograms*

Description

Decorate Heatmap Dendrograms

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 dendrogram.
`slice`	Index of the row slice or column slice in the heatmap.
`which`	Is the dendrogram on rows or on columns?
`envir`	Where to look for variables inside `code`.

Details

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

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

Value

This function returns no value.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-decoration.html

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 *Decorate Heatmap Dimension Names*

Description

Decorate 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 viewport.
`slice`	Index of the row slice or column slice in the heatmap.
`which`	on rows or on columns?
`envir`	where to look for variables inside code.

Details

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^th name is located at:

```
    # assume nc is the number of columns in the column slice
    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 Heatmap Bodies*

Description

Decorate Heatmap Bodies

Usage

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

Arguments

`heatmap`	Name of the heatmap which is set as `name` argument in `Heatmap` function.
`code`	Code that adds graphics in the selected heatmap body.
`slice`	Index of the row slice in the heatmap.
`row_slice`	Index of the row slice in the heatmap.
`column_slice`	Index of the column slice in the heatmap.
`envir`	Where to look for variables inside `code`.

Details

There is a viewport for each slice in each heatmap. This function contructs the name of the viewport, goes to the viewport by `seekViewport`, runs the code to that viewport and finally goes back to the original viewport.

Value

This function returns no value.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-decoration.html`

Examples

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

`decorate_row_dend` *Decorate Heatmap Row Dendrograms*

Description

Decorate Heatmap Row Dendrograms

Usage

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

Arguments

`...`	Pass to `decorate_dend`.
`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

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

`decorate_row_names` *Decorate Heatmap Row Names*

Description

Decorate Heatmap Row Names

Usage

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

Arguments

`...`	Pass to `decorate_dimnames`.
`envir`	wWhere 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

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

`decorate_row_title` *Decorate Heatmap Row Titles*

Description

Decorate Heatmap Row Titles

Usage

```
decorate_row_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

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

`decorate_title` *Decorate Heatmap Titles*

Description

Decorate Heatmap Titles

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 viewport.
`slice`	Index of the row slice or column slice in the heatmap.
`which`	Is it a row title or a column title?
`envir`	Where to look for variables inside `code`.

Details

There is a viewport for row titles and column title in the heatmap. This function contructs the name of the viewport, goes to the viewport by `seekViewport` , runs code to that viewport and finally goes back to the original viewport.

Value

The function returns no value.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-decoration.html`

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)
```

`default_axis_param` *The Default Parameters for Annotation Axis*

Description

The Default Parameters for Annotation Axis

Usage

```
default_axis_param(which)
```

Arguments

`which`	Whether it is for column annotation or row annotation?

Details

There are following parameters for the annotation axis:

at The breaks of axis. By default it is automatically inferred.

labels The corresponding axis labels.

labels_rot The rotation of the axis labels.

gp Graphc parameters of axis labels. The value should be a `unit` object.

side If it is for column annotation, the value should only be one of `left` and `right`. If it is for row annotation, the value should only be one of `top` and `bottom`.

facing Whether the axis faces to the outside of the annotation region or inside. Sometimes when appending more than one heatmaps, the axes of column annotations of one heatmap might overlap to the neighbouring heatmap, setting `facing` to `inside` may invoild it.

direction The direction of the axis. Value should be "normal" or "reverse".

All the parameters are passed to `annotation_axis_grob` to construct an axis grob.

Examples

```
default_axis_param("column")
default_axis_param("row")
```

`default_get_type` *Default get_type for oncoPrint()*

Description

Default get_type for oncoPrint()

Usage

```
default_get_type(x)
```

Arguments

`x`	A strings which encode multiple altertations.

Details

It recognizes following separators: `;:,|`.

Examples

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

`dendrogramGrob` *Grob for Dendrogram*

Description

Grob for Dendrogram

Usage

```
dendrogramGrob(dend, facing = c("bottom", "top", "left", "right"),
    order = c("normal", "reverse"), gp = gpar())
```

Arguments

`dend`	A `dendrogram` object.
`facing`	Facing of the dendrogram.
`order`	If it is set to `reverse`, the first leaf is put on the right if the dendrogram is horizontal and it is put on the top if the dendrogram is vertical.
`gp`	Graphic parameters for the dendrogram segments. If any of `col`, `lwd` or `lty` is set in the `edgePar` attribute of a node, the corresponding value defined in `gp` will be overwritten for this node, so `gp` is like global graphic parameters for dendrogram segments.

Details

If dend has not been processed by `adjust_dend_by_x`, internally `adjust_dend_by_x` is called to add `x` attributes to each node/leaf.

Value

A `grob` object which is contructed by `segmentsGrob`.

Examples

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

dend_heights *Height of the Dendrograms*

Description

Height of the Dendrograms

Usage

```
dend_heights(x)
```

Arguments

x	a dendrogram object or a list of dendrogram objects.

Examples

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

dend_xy *Coordinates of the Dendrogram*

Description

Coordinates of the Dendrogram

Usage

```
dend_xy(dend)
```

Arguments

dend	a dendrogram object.

Details

dend will be processed by adjust_dend_by_x if it is processed yet.

Value

A list of leave positions (x) and dendrogram height (y).

Examples

```
m = matrix(rnorm(100), 10)
dend1 = as.dendrogram(hclust(dist(m)))
dend_xy(dend1)

dend1 = adjust_dend_by_x(dend1, sort(runif(10)))
dend_xy(dend1)

dend1 = adjust_dend_by_x(dend1, unit(1:10, "cm"))
dend_xy(dend1)
```

---

densityHeatmap *Visualize Density Distribution by Heatmap*

Description

Visualize Density Distribution by Heatmap

Usage

```
densityHeatmap(data,
    density_param = list(na.rm = TRUE),

    col = rev(brewer.pal(11, "Spectral")),
    color_space = "LAB",
    ylab = deparse(substitute(data)),
    column_title = paste0("Density heatmap of ", deparse(substitute(data))),
    title = column_title,
    ylim = NULL,
    range = ylim,

    title_gp = gpar(fontsize = 14),
    ylab_gp = gpar(fontsize = 12),
    tick_label_gp = gpar(fontsize = 10),
    quantile_gp = gpar(fontsize = 10),
    show_quantiles = TRUE,

    column_order = NULL,
    column_names_side = "bottom",
    show_column_names = TRUE,
    column_names_max_height = unit(6, "cm"),
    column_names_gp = gpar(fontsize = 12),
    column_names_rot = 90,

    cluster_columns = FALSE,
    clustering_distance_columns = "ks",
    clustering_method_columns = "complete",
    mc.cores = 1, cores = mc.cores,
    ...)
```

Arguments

data	A matrix or a list. If it is a matrix, density is calculated by columns.
density_param	Parameters send to density, na.rm is enforced to be TRUE.
col	A vector of colors that density values are mapped to.
color_space	The color space in which colors are interpolated. Pass to colorRamp2.
ylab	Label on y-axis.
column_title	Title of the heatmap.
title	Same as column_title.
ylim	Ranges on the y-axis.
range	Same as ylim.
title_gp	Graphic parameters for title.
ylab_gp	Graphic parameters for y-labels.
tick_label_gp	Graphic parameters for y-ticks.
quantile_gp	Graphic parameters for the quantiles.
show_quantiles	Whether show quantile lines.
column_order	Order of columns.
column_names_side	Pass to Heatmap.
show_column_names	Pass to Heatmap.
column_names_max_height	Pass to Heatmap.
column_names_gp	Pass to Heatmap.
column_names_rot	Pass to Heatmap.
cluster_columns	Whether cluster columns?
clustering_distance_columns	There is a specific distance method ks which is the Kolmogorov-Smirnov statistic between two distributions. For other methods, the distance is calculated on the density matrix.
clustering_method_columns	Pass to Heatmap.
mc.cores	Multiple cores for calculating ks distance. This argument will be removed in future versions.
cores	Multiple cores for calculating ks distance.
...	Pass to Heatmap.

Details

To visualize data distribution in a matrix or in a list, we normally use boxplot or violinplot. We can also use colors to map the density values and visualize distribution of values 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 maximun and minimal values in all densities.

Value

A `Heatmap-class` object. It can oly add other heatmaps/annotations vertically.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/other-high-level-plots.html#density-heatmap

Examples

```
matrix = matrix(rnorm(100), 10); colnames(matrix) = letters[1:10]
densityHeatmap(matrix)

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

ha = HeatmapAnnotation(points = anno_points(runif(10)),
    anno = rep(c("A", "B"), each = 5), col = list(anno = c("A" = "red", "B" = "blue")))
densityHeatmap(matrix, top_annotation = ha)
densityHeatmap(matrix, top_annotation = ha) %v% Heatmap(matrix, height = unit(6, "cm"))
```

`dim.Heatmap` *Dimension of the Heatmap*

Description

Dimension of the Heatmap

Usage

```
## S3 method for class 'Heatmap'
dim(x)
```

Arguments

`x`	A `Heatmap-class` object.

Examples

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

---

dist2 *Calculate Pairwise Distance from a Matrix*

Description

Calculate Pairwise Distance from a Matrix

Usage

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

Arguments

`x`	A matrix or a list. If it is a matrix, the distance is calculated by rows.
`pairwise_fun`	A function which calculates distance between two vectors.
`...`	Pass to `as.dist`.

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

```
lt = lapply(1:10, function(i) {
    sample(letters, sample(6:10, 1))
})
dist2(lt, function(x, y) {
    length(intersect(x, y))/length(union(x, y))
})
```

draw-AnnotationFunction-method

Draw the AnnotationFunction Object

Description

Draw the AnnotationFunction Object

Usage

```
## S4 method for signature 'AnnotationFunction'
draw(object, index, k = 1, n = 1, test = FALSE, ...)
```

Arguments

`object`	The `AnnotationFunction-class` object.
`index`	Index of observations.
`k`	Current slice index.
`n`	Total number of slices.
`test`	Is it in test mode? The value can be logical or a text which is plotted as the title of plot.
`...`	Pass to `viewport`.

Details

Normally it is called internally by the `SingleAnnotation-class`.

When `test` is set to `TRUE`, the annotation graphic is directly drawn, which is generally for testing purpose.

Examples

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

draw-dispatch *Method dispatch page for draw*

Description

Method dispatch page for `draw`.

Dispatch

`draw` can be dispatched on following classes:

- `draw,AnnotationFunction-method`, `AnnotationFunction-class` class method
- `draw,SingleAnnotation-method`, `SingleAnnotation-class` class method
- `draw,HeatmapAnnotation-method`, `HeatmapAnnotation-class` class method
- `draw,Heatmap-method`, `Heatmap-class` class method
- `draw,HeatmapList-method`, `HeatmapList-class` class method
- `draw,Legends-method`, `Legends-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 `Heatmap-class` object.
`internal`	If `TRUE`, it is only used inside the calling of `draw,HeatmapList-method`. It only draws the heatmap without legends where the legend will be drawn by `draw,HeatmapList-method`.
`test`	Only for testing. If it is `TRUE`, the heatmap body is directly drawn.
`...`	Pass to `draw,HeatmapList-method`.

Details

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

There are some arguments which control the some settings of the heatmap such as legends. Please go to draw,HeatmapList-method for these arguments.

Value

A HeatmapList-class object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

---

draw-HeatmapAnnotation-method
Draw the Heatmap Annotations

Description

Draw the Heatmap Annotations

Usage

```
## S4 method for signature 'HeatmapAnnotation'
draw(object, index, k = 1, n = 1, ...,
    test = FALSE, anno_mark_param = list())
```

Arguments

`object`	A HeatmapAnnotation-class object.
`index`	A vector of indices.
`k`	The current slice index for the annotation if it is split.
`n`	Total number of slices.
`...`	Pass to viewport which contains all the annotations.
`test`	Is it in test mode? The value can be logical or a text which is plotted as the title of plot.
`anno_mark_param`	It contains specific parameters for drawing anno_mark and pass to the draw,SingleAnnotation-method.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

---

draw-HeatmapList-method

Draw a list of heatmaps

Description

Draw a list of heatmaps

Usage

```
## S4 method for signature 'HeatmapList'
draw(object,
    newpage = TRUE,
    background = "white",

    row_title = character(0),
    row_title_side = c("left", "right"),
    row_title_gp = gpar(fontsize = 13),
    column_title = character(0),
    column_title_side = c("top", "bottom"),
    column_title_gp = gpar(fontsize = 13),

    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(),
    align_heatmap_legend = NULL,
    align_annotation_legend = NULL,
    legend_grouping = c("adjusted", "original"),

    gap = unit(2, "mm"),
    ht_gap = gap,

    main_heatmap = which(sapply(object@ht_list, inherits, "Heatmap"))[1],
    padding = GLOBAL_PADDING,
    adjust_annotation_extension = NULL,

    auto_adjust = TRUE,
    row_dend_side = c("original", "left", "right"),
    row_sub_title_side = c("original", "left", "right"),
    column_dend_side = c("original", "top", "bottom"),
    column_sub_title_side = c("original", "top", "bottom"),

    row_gap = NULL,
    cluster_rows = NULL,
    cluster_row_slices = 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,
    row_km = km,
    row_km_repeats = NULL,
    row_split = split,
    height = NULL,
    heatmap_height = NULL,

    column_gap = NULL,
    cluster_columns = NULL,
    cluster_column_slices = NULL,
    clustering_distance_columns = NULL,
    clustering_method_columns = NULL,
    column_dend_width = NULL,
    show_column_dend = NULL,
    column_dend_reorder = NULL,
    column_dend_gp = NULL,
    column_order = NULL,
    column_km = NULL,
    column_km_repeats = NULL,
    column_split = NULL,
    width = NULL,
    heatmap_width = NULL,

    use_raster = NULL,
    raster_device = NULL,
    raster_quality = NULL,
    raster_device_param = NULL,
    raster_resize = NULL,

    post_fun = NULL,
    save_last = ht_opt$save_last,

    ### global setting
    heatmap_row_names_gp = NULL,
    heatmap_column_names_gp = NULL,
    heatmap_row_title_gp = NULL,
    heatmap_column_title_gp = NULL,
    legend_title_gp = NULL,
    legend_title_position = NULL,
    legend_labels_gp = NULL,
    legend_grid_height = NULL,
    legend_grid_width = NULL,
    legend_border = NULL,
    legend_gap = NULL,
    heatmap_border = NULL,
    annotation_border = NULL,
    fastcluster = NULL,
    simple_anno_size = NULL,
    show_parent_dend_line = NULL)
```

Arguments

`object`	a `HeatmapList-class` object.
`newpage`	whether create a new page for the graphics. If you want to arrange multiple plots in one page, I suggest to use `grid.grabExpr`.
`background`	Background color of the whole plot.
`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 to put heatmap legend
`merge_legends`	merge heatmap legends and annotation legends to put into one column.
`show_heatmap_legend`	whether show all heatmap legends
`heatmap_legend_list`	use-defined legends which are put after the heatmap legends

`annotation_legend_side`	side of the annotation legends
`show_annotation_legend`	whether show annotation legends
`annotation_legend_list`	user-defined legends which are put after the annotation legends
`align_heatmap_legend`	How to align the legends to heatmap. Possible values are "heatmap_center", "heatmap_top" and "global_center". If the value is `NULL`, it automatically picks the proper value from the three options.
`align_annotation_legend`	How to align the legends to heatmap. Possible values are "heatmap_center", "heatmap_top" and "global_center".
`legend_grouping`	How the legends are grouped. Values should be "adjusted" or "original". If it is set as "original", all annotation legends are grouped together.
`gap`	gap between heatmaps/annotations
`ht_gap`	same as `gap`.
`main_heatmap`	index of main heatmap. The value can be a numeric index or the heatmap name
`padding`	padding of the whole plot. The value is a unit vector of length 4, which corresponds to bottom, left, top and right.
`adjust_annotation_extension`	whether take annotation name into account when calculating positions of graphic elements.
`auto_adjust`	whether apply automatic adjustment? The auto-adjustment includes turning off dendrograms, titles and row/columns for non-main heatmaps.
`row_dend_side`	side of the dendrogram from the main heatmap
`row_sub_title_side`	side of the row title from the main heatmap
`column_dend_side`	side of the dendrogram from the main heatmap
`column_sub_title_side`	side of the column title from the main heatmap
`row_gap`	this modifies `row_gap` of the main heatmap
`cluster_rows`	this modifies `cluster_rows` of the main heatmap
`cluster_row_slices`	this modifies `cluster_row_slices` of the main heatmap
`clustering_distance_rows`	this modifies `clustering_distance_rows` of the main heatmap
`clustering_method_rows`	this modifies `clustering_method_rows` of the main heatmap
`row_dend_width`	this modifies `row_dend_width` of the main heatmap
`show_row_dend`	this modifies `show_row_dend` of the main heatmap

`row_dend_reorder`	this modifies `row_dend_reorder` of the main heatmap
`row_dend_gp`	this modifies `row_dend_gp` of the main heatmap
`row_order`	this modifies `row_order` of the main heatmap
`km`	= this modifies `km` of the main heatmap
`split`	this modifies `split` of the main heatmap
`row_km`	this modifies `row_km` of the main heatmap
`row_km_repeats`	this modifies `row_km_repeats` of the main heatmap
`row_split`	this modifies `row_split` of the main heatmap
`height`	this modifies `height` of the main heatmap
`heatmap_height`	this modifies `heatmap_height` of the main heatmap
`column_gap`	this modifies `column_gap` of the main heatmap
`cluster_columns`	this modifies `cluster_columns` of the main heatmap
`cluster_column_slices`	this modifies `cluster_column_slices` of the main heatmap
`clustering_distance_columns`	this modifies `clustering_distance_columns` of the main heatmap
`clustering_method_columns`	this modifies `clustering_method_columns` of the main heatmap
`column_dend_width`	this modifies `column_dend_width` of the main heatmap
`show_column_dend`	this modifies `show_column_dend` of the main heatmap
`column_dend_reorder`	this modifies `column_dend_reorder` of the main heatmap
`column_dend_gp`	this modifies `column_dend_gp` of the main heatmap
`column_order`	this modifies `column_order` of the main heatmap
`column_km`	this modifies `column_km` of the main heatmap
`column_km_repeats`	this modifies `column_km_repeats` of the main heatmap
`column_split`	this modifies `column_split` of the main heatmap
`width`	this modifies `width` of the main heatmap
`heatmap_width`	this modifies `heatmap_width` of the main heatmap
`use_raster`	this modifies `use_raster` of every heatmap.
`raster_device`	this modifies `raster_device` of every heatmap.
`raster_quality`	this modifies `raster_quality` of every heatmap.
`raster_device_param`	this modifies `raster_device_param` of every heatmap.
`raster_resize`	this modifies `raster_resize` of every heatmap.

`post_fun`	A self-defined function will be executed after all the heatmaps are drawn.
`save_last`	Whether to save the last plot?
`heatmap_row_names_gp`	this set the value in `ht_opt` and reset back after the plot is done
`heatmap_column_names_gp`	this set the value in `ht_opt` and reset back after the plot is done
`heatmap_row_title_gp`	this set the value in `ht_opt` and reset back after the plot is done
`heatmap_column_title_gp`	this set the value in `ht_opt` and reset back after the plot is done
`legend_title_gp`	this set the value in `ht_opt` and reset back after the plot is done
`legend_title_position`	this set the value in `ht_opt` and reset back after the plot is done
`legend_labels_gp`	this set the value in `ht_opt` and reset back after the plot is done
`legend_grid_height`	this set the value in `ht_opt` and reset back after the plot is done
`legend_grid_width`	this set the value in `ht_opt` and reset back after the plot is done
`legend_border`	this set the value in `ht_opt` and reset back after the plot is done
`legend_gap`	Gap between legends. The value should be a vector of two units. One for gaps between vertical legends and one for the horizontal legends. If only one single unit is specified, the same gap set for the vertical and horizontal legends.
`heatmap_border`	this set the value in `ht_opt` and reset back after the plot is done
`annotation_border`	this set the value in `ht_opt` and reset back after the plot is done
`fastcluster`	this set the value in `ht_opt` and reset back after the plot is done
`simple_anno_size`	this set the value in `ht_opt` and reset back after the plot is done
`show_parent_dend_line`	this set the value in `ht_opt` and reset back after the plot is done

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 `HeatmapList-class` object for which the layout has been created.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/a-list-of-heatmaps.html

Examples

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

---

draw-Legends-method *Draw the Legends*

Description

Draw the Legends

Usage

```
## S4 method for signature 'Legends'
draw(object, x = unit(0.5, "npc"), y = unit(0.5, "npc"), just = "centre", test = FALSE)
```

Arguments

`object`	The `grob` object returned by `Legend` or `packLegend`.
`x`	The x position of the legends, measured in current viewport.
`y`	The y position of the legends, measured in current viewport.
`just`	Justification of the legends.
`test`	Only used for testing.

Details

In the legend grob, there should always be a viewport attached which is like a wrapper of all the graphic elements in a legend. If in the `object`, there is already a viewport attached, it will modify the `x`, `y` and `valid.just` of the viewport. If there is not viewport attached, a viewport with specified `x`, `y` and `valid.just` is created and attached.

You can also directly use `grid.draw` to draw the legend object, but you can only control the position of the legends by first creating a parent viewport and adjusting the position of the parent viewport.

Examples

```
lgd = Legend(at = 1:4, title = "foo")
draw(lgd, x = unit(0, "npc"), y = unit(0, "npc"), just = c("left", "bottom"))

# and a similar version of grid.draw
pushViewport(viewport(x = unit(0, "npc"), y = unit(0, "npc"), just = c("left", "bottom")))
grid.draw(lgd)
popViewport()
```

`draw-SingleAnnotation-method`
Draw the Single Annotation

Description

Draw the Single Annotation

Usage

```
## S4 method for signature 'SingleAnnotation'
draw(object, index, k = 1, n = 1, test = FALSE,
    anno_mark_param = list())
```

Arguments

`object`	A `SingleAnnotation-class` object.
`index`	A vector of indices.
`k`	The index of the slice.
`n`	Total number of slices. `k` and `n` are used to adjust annotation names. E.g. if `k` is 2 and `n` is 3, the annotation names are not drawn.
`test`	Is it in test mode? The value can be logical or a text which is plotted as the title of plot.
`anno_mark_param`	It contains specific parameters for drawing `anno_mark`.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

draw_annotation-Heatmap-method

Draw Heatmap Annotations on the Heatmap

Description

Draw Heatmap Annotations on the Heatmap

Usage

```
## S4 method for signature 'Heatmap'
draw_annotation(object, which = c("top", "bottom", "left", "right"), k = 1, ...)
```

Arguments

`object`	A `Heatmap-class` object.
`which`	The position of the heamtap annotation.
`k`	Slice index.
`...`	Pass to `viewport` which includes the complete heatmap annotation.

Details

A viewport is created which contains column/top annotations.

The function calls `draw,HeatmapAnnotation-method` 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

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

draw_annotation_legend-HeatmapList-method

Draw legends for All Annotations

Description

Draw legends for All Annotations

Usage

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

Arguments

`object`	A `HeatmapList-class` object.
`legend_list`	A list of self-defined legends, should be wrapped into `grob` objects. It is normally constructed by `Legend`.
`...`	Other arguments.

Details

We call the "annotation legends" as the secondary legends. For horizontal heamtap list, the legends are those from all top/bottom annotations, and for vertical heatmap list, the legends are those from all left/right annotations.

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

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

draw_dend-Heatmap-method

Draw Heatmap Dendrograms

Description

Draw Heatmap Dendrograms

Usage

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

Arguments

object	A Heatmap-class object.
which	Are the dendrograms put on the row or on the column of the heatmap?
k	Slice index.
max_height	maximal height of dendrogram.
...	Pass to viewport which includes the complete heatmap dendrograms.

Details

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, ...)
```

Arguments

object	A Heatmap-class object.
which	Are the names put on the row or on the column of the heatmap?
k	Slice index.
...	Pass to viewport which includes the complete heatmap row/column names.

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

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

draw_heatmap_body-Heatmap-method

Draw Heatmap Body

Description

Draw Heatmap Body

Usage

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

Arguments

`object`	A `Heatmap-class` object.
`kr`	Row slice index.
`kc`	Column slice index.
`...`	Pass to `viewport` which includes the slice of heatmap body.

Details

A viewport is created which contains subset rows and columns 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

```
# There is no example
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

`object`	A `HeatmapList-class` object.
`legend_list`	A list of self-defined legends, should be wrapped into `grob` objects. It is normally constructed by `Legend`.
`...`	Other arguments.

Details

Actually we call the "heatmap legends" as the main legends. For horizontal heatmap list, the legends are those from heamtap/row annotation/left/right annotation. For vertical heatmap list, the legends are those from heamtap/column annotation/top/bottom annotation. if `merge_legends` is true in `draw,HeatmapList-method`, then it contains all legends shown on the plot.

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

```
# There is no example
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

It only draws the list of heatmaps without legends and titles.

This function is only for internal use.

Value

This function returns no value.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
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 `Heatmap-class` object.
`which`	Is title put on the row or on the column of the heatmap?
`k`	Slice index.
`...`	Pass to `viewport` which includes the complete heatmap title.

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

```
# There is no example
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	Is it a row title or a column title.

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

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

---

extract_comb *Extract Elements in a Combination set*

Description

Extract Elements in a Combination set

Usage

```
extract_comb(m, comb_name)
```

Arguments

m	A combination matrix returned by `make_comb_mat`.
comb_name	The valid combination set name should be from `comb_name`.

Details

It returns the combination set.

Examples

```
set.seed(123)
lt = list(a = sample(letters, 10),
          b = sample(letters, 15),
          c = sample(letters, 20))
m = make_comb_mat(lt)
extract_comb(m, "110")
```

---

frequencyHeatmap *Visualize Frequency Distribution by Heatmap*

Description

Visualize Frequency Distribution by Heatmap

Usage

```
frequencyHeatmap(data,
    breaks = "Sturges",
    stat = c("count", "density", "proportion"),

    col = brewer.pal(9, "Blues"),
    color_space = "LAB",
    ylab = deparse(substitute(data)),
    column_title = paste0("Frequency heatmap of ", deparse(substitute(data))),
    title = column_title,
    ylim = NULL,
    range = ylim,

    title_gp = gpar(fontsize = 14),
    ylab_gp = gpar(fontsize = 12),
    tick_label_gp = gpar(fontsize = 10),

    column_order = NULL,
    column_names_side = "bottom",
    show_column_names = TRUE,
    column_names_max_height = unit(6, "cm"),
    column_names_gp = gpar(fontsize = 12),
    column_names_rot = 90,
    cluster_columns = FALSE,

    use_3d = FALSE,
    ...)
```

Arguments

`data`	A matrix or a list. If it is a matrix, density is calculated by columns.
`breaks`	Pass to `hist`. Please only set equal bin size.
`stat`	Statistic to use.
`col`	A vector of colors that density values are mapped to.
`color_space`	The color space in which colors are interpolated. Pass to `colorRamp2`.
`ylab`	Label on y-axis.
`column_title`	Title of the heatmap.
`title`	Same as `column_title`.
`ylim`	Ranges on the y-axis.
`range`	Same as `ylim`.
`title_gp`	Graphic parameters for title.
`ylab_gp`	Graphic parameters for y-labels.
`tick_label_gp`	Graphic parameters for y-ticks.
`column_order`	Order of columns.

`column_names_side`	Pass to `Heatmap`.
`show_column_names`	Pass to `Heatmap`.
`column_names_max_height`	Pass to `Heatmap`.
`column_names_gp`	Pass to `Heatmap`.
`column_names_rot`	Pass to `Heatmap`.
`cluster_columns`	Whether cluster columns?
`use_3d`	Whether to visualize the frequencies as a 3D heatmap with `Heatmap3D`?
`...`	Pass to `Heatmap` or `Heatmap3D` (if `use_3d = TRUE`).

Value

A `Heatmap-class` object. It can oly add other heatmaps/annotations vertically.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
matrix = matrix(rnorm(100), 10); colnames(matrix) = letters[1:10]
frequencyHeatmap(matrix)
frequencyHeatmap(matrix, use_3d = TRUE)
```

`full_comb_code` *Full set of code of combination sets*

Description

Full set of code of combination sets

Usage

```
full_comb_code(n, complement = FALSE)
```

Arguments

`n`	Number of sets
`complement`	Whether include the code for complement set?

Examples

```
full_comb_code(2)
full_comb_code(3)
full_comb_code(4)
full_comb_code(4, TRUE)
```

`getXY_in_parent_vp` *Convert XY in a Parent Viewport*

Description

Convert XY in a Parent Viewport

Usage

```
getXY_in_parent_vp(u, vp_name = "ROOT")
```

Arguments

u	A list of two units which correspond to x and y.
vp_name	The name of the parent viewport.

Details

It converts a coordinate measured in current viewport to the coordinate in a parent viewport.

In the conversion, all units are recalculated as absolute units, so if you change the size of the interactive graphic window, you need to rerun the function.

Value

A list of two units.

Examples

```
grid.newpage()
pushViewport(viewport(x = 0.5, y = 0.5, width = 0.5, height = 0.5, just = c("left", "bottom")))
grid.rect()
grid.points(x = unit(2, "cm"), y = unit(2, "cm"), pch = 1)
u = list(x = unit(2, "cm"), y = unit(2, "cm"))
u2 = getXY_in_parent_vp(u)
popViewport()
grid.rect(gp = gpar(col = "red"))
grid.points(x = u2$x, u2$y, pch = 2)
```

`get_color_mapping_list-HeatmapAnnotation-method` *Get a List of ColorMapping objects*

Description

Get a List of ColorMapping objects

Usage

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

Arguments

`object`	A `HeatmapAnnotation-class` object.

Details

Color mappings 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

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

`get_legend_param_list-HeatmapAnnotation-method` *Get a List of Annotation Legend Parameters*

Description

Get a List of Annotation Legend Parameters

Usage

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

Arguments

`object`	A `HeatmapAnnotation-class` object.

Details

The annotation legend 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.annotation_axis` *Draw Annotation Axis*

Description

Draw Annotation Axis

Usage

```
grid.annotation_axis(at = NULL, labels = at, labels_rot = 0, gp = gpar(),
    side = "left", facing = "outside", direction = "normal")
```

Arguments

`at`	Break values. If it is not specified, it is inferred from data scale in current viewport.
`labels`	Corresponding labels.
`labels_rot`	Rotations of labels.
`gp`	Graphic parameters.
`side`	side of the axis of the annotation viewport.
`facing`	Facing of the axis.
`direction`	direction of the axis. Value should be "normal" or "reverse".

Details

It uses `annotation_axis_grob` to construct the grob object, then use `grid.draw` to draw the axis.

Examples

```
# See examples in `annotation_axis_grob`
NULL
```

`grid.boxplot` *Draw a Single Boxplot*

Description

Draw a Single Boxplot

Usage

```
grid.boxplot(value, pos, outline = TRUE, box_width = 0.6,
    pch = 1, size = unit(2, "mm"), gp = gpar(fill = "#CCCCCC"),
    direction = c("vertical", "horizontal"))
```

Arguments

`value`	A vector of numeric values.
`pos`	Position of the boxplot.
`outline`	Whether draw outlines?
`box_width`	width of the box.
`pch`	Point type.
`size`	Point size.
`gp`	Graphic parameters.
`direction`	Whether the box is vertical or horizontal.

Details

All the values are measured with `native` coordinate.

Examples

```
lt = list(rnorm(100), rnorm(100))
grid.newpage()
pushViewport(viewport(xscale = c(0.5, 2.5), yscale = range(lt)))
grid.boxplot(lt[[1]], pos = 1, gp = gpar(fill = "red"))
grid.boxplot(lt[[2]], pos = 2, gp = gpar(fill = "green"))
popViewport()
```

`grid.dendrogram` *Draw the Dendrogram*

Description

Draw the Dendrogram

Usage

```
grid.dendrogram(dend, ..., test = FALSE)
```

Arguments

`dend`	A `dendrogram` object.
`...`	Pass to `dendrogramGrob`.
`test`	Is it in test mode? If it is in test mode, a viewport is created by calculating proper xlim and ylim.

Details

`grid.dendrogram` supports drawing dendrograms with self-defind leaf positions. The positions of leaves can be defined by `adjust_dend_by_x`. Also the dendrogram can be customized by setting the edgePar attribute for each node (basically for controlling the style of segments), e.g. by `color_branches`.

To draw the dendrogram, a viewport should be firstly created. `dend_xy` can be used to get the positions of leaves and height of the dendrogram.

Examples

```
m = matrix(rnorm(100), 10)
dend = as.dendrogram(hclust(dist(m)))
grid.newpage()
pushViewport(viewport(xscale = c(0, 10.5), yscale = c(0, dend_heights(dend)),
    width = 0.9, height = 0.9))
grid.dendrogram(dend)
```

```
popViewport()

grid.dendrogram(dend, test = TRUE)

require(dendextend)
dend = color_branches(dend, k = 2)
dend = adjust_dend_by_x(dend, unit(sort(runif(10)*10), "cm"))
grid.dendrogram(dend, test = TRUE)
```

grid.draw.Legends *Draw the Legends*

Description

Draw the Legends

Usage

```
## S3 method for class 'Legends'
grid.draw(x, recording = TRUE)
```

Arguments

x	The grob object returned by Legend or packLegend.
recording	Pass to grid.draw.

Details

This function is actually an S3 method of the Legends class for the grid.draw general method. It applies grid.draw on the grob slot of the object.

Examples

```
lgd = Legend(at = 1:4, title = "foo")
pushViewport(viewport(x = unit(0, "npc"), y = unit(0, "npc"), just = c("left", "bottom")))
grid.draw(lgd)
popViewport()
```

gt_render *Mark the text for the rendering by gridtext package*

Description

Mark the text for the rendering by gridtext package

Usage

```
gt_render(x, ...)
```

Arguments

`x`	Text labels. The value can be a vector.
`...`	Other parameters passed to `richtext_grob`.

Details

Text marked by `gt_render` will be rendered by `richtext_grob` function.

Examples

```
if(requireNamespace("gridtext")) {
mat = matrix(rnorm(100), 10)
rownames(mat) = letters[1:10]
ht = Heatmap(mat,
column_title = gt_render("Some <span style='color:blue'>blue text **in bold.**</span><br>And *italics text.*<br>A
column_title_gp = gpar(box_fill = "orange"),
row_labels = gt_render(letters[1:10], padding = unit(c(2, 10, 2, 10), "pt")),
row_names_gp = gpar(box_col = "red"),
row_km = 2,
row_title = gt_render(c("title1", "title2")),
row_title_gp = gpar(box_fill = "yellow"),
heatmap_legend_param = list(
title = gt_render("<span style='color:orange'>**Legend title**</span>"),
title_gp = gpar(box_fill = "grey"),
at = c(-3, 0, 3),
labels = gt_render(c("*negative* three", "zero", "*positive* three"))
))
ht = rowAnnotation(
foo = anno_text(gt_render(sapply(LETTERS[1:10], strrep, 10), align_widths = TRUE),
                gp = gpar(box_col = "blue", box_lwd = 2),
                just = "right",
                location = unit(1, "npc")
)) + ht
draw(ht)

}
```

`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),
    border = NA,
    border_gp = gpar(col = "black"),
    cell_fun = NULL,
    layer_fun = NULL,
    jitter = FALSE,

    row_title = character(0),
    row_title_side = c("left", "right"),
    row_title_gp = gpar(fontsize = 13.2),
    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 = 13.2),
    column_title_rot = 0,

    cluster_rows = TRUE,
    cluster_row_slices = 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 = is.logical(cluster_rows) || is.function(cluster_rows),
    row_dend_gp = gpar(),
    cluster_columns = TRUE,
    cluster_column_slices = 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 = is.logical(cluster_columns) || is.function(cluster_columns),
    row_order = NULL,
    column_order = NULL,

    row_labels = rownames(matrix),
    row_names_side = c("right", "left"),
    show_row_names = TRUE,
    row_names_max_width = unit(6, "cm"),
    row_names_gp = gpar(fontsize = 12),
    row_names_rot = 0,
    row_names_centered = FALSE,
    column_labels = colnames(matrix),
    column_names_side = c("bottom", "top"),
    show_column_names = TRUE,
    column_names_max_height = unit(6, "cm"),
    column_names_gp = gpar(fontsize = 12),
    column_names_rot = 90,
    column_names_centered = FALSE,

    top_annotation = NULL,
    bottom_annotation = NULL,
    left_annotation = NULL,
    right_annotation = NULL,

    km = 1,
    split = NULL,
    row_km = km,
    row_km_repeats = 1,
    row_split = split,
    column_km = 1,
    column_km_repeats = 1,
    column_split = NULL,
    gap = unit(1, "mm"),
    row_gap = unit(1, "mm"),
    column_gap = unit(1, "mm"),
    show_parent_dend_line = ht_opt$show_parent_dend_line,

    heatmap_width = unit(1, "npc"),
    width = NULL,
    heatmap_height = unit(1, "npc"),
    height = NULL,

    show_heatmap_legend = TRUE,
    heatmap_legend_param = list(title = name),

    use_raster = NULL,
   raster_device = c("png", "jpeg", "tiff", "CairoPNG", "CairoJPEG", "CairoTIFF", "agg_png"),
    raster_quality = 1,
    raster_device_param = list(),
    raster_resize_mat = FALSE,
    raster_by_magick = requireNamespace("magick", quietly = TRUE),
    raster_magick_filter = NULL,

    post_fun = NULL)
```

Arguments

matrix	A matrix. Either numeric or character. If it is a simple vector, it will be converted to a one-column matrix.
col	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 `colorRamp2`). If the matrix is continuous, the value can also be a vector of colors so that colors can be interpolated. Pass to `ColorMapping`. For more details and examples, please refer to `https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#colors` .
name	Name of the heatmap. By default the heatmap name is used as the title of the heatmap legend.
na_col	Color for `NA` values.
rect_gp	Graphic parameters for drawing rectangles (for heatmap body). The value should be specified by `gpar` and `fill` parameter is ignored.
color_space	The color space in which colors are interpolated. Only used if `matrix` is numeric and `col` is a vector of colors. Pass to `colorRamp2`.
border	Whether draw border. The value can be logical or a string of color.
border_gp	Graphic parameters for the borders. If you want to set different parameters for different heatmap slices, please consider to use `decorate_heatmap_body`.
cell_fun	Self-defined function to add graphics on each cell. Seven parameters will be passed into this function: `j`, `i`, `x`, `y`, `width`, `height`, `fill` which are column index, row index in `matrix`, coordinate of the cell, the width and height of the cell and the filled color. `x`, `y`, `width` and `height` are all `unit` objects.
layer_fun	Similar as `cell_fun`, but is vectorized. Check `https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#customize-the-heatmap-body` .
jitter	Random shifts added to the matrix. The value can be logical or a single numeric value. It it is `TRUE`, random values from uniform distribution between 0 and 1e-10 are generated. If it is a numeric value, the range for the uniform distribution is (0, `jitter`). It is mainly to solve the problem of "Error: node stack overflow" when there are too many identical rows/columns for plotting the dendrograms. ADD: From version 2.5.6, the error of node stack overflow has been fixed, now this argument is ignored.
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 row title.
row_title_rot	Rotation of row title. Only 0, 90, 270 are allowed to set.

`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 column title.
`column_title_rot`	Rotation of column titles. Only 0, 90, 270 are allowed to set.
`cluster_rows`	If the value is a logical, it controls whether to make cluster on rows. The value can also be a `hclust` or a `dendrogram` which already contains clustering. Check `https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#clustering` .
`cluster_row_slices`	If rows are split into slices, whether perform clustering on the slice means?
`clustering_distance_rows`	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 `dist` object. If the function has two arguments, the input arguments are two vectors and the function calculates distance between these two vectors.
`clustering_method_rows`	Method to perform hierarchical clustering, pass to `hclust`.
`row_dend_side`	Should the row dendrogram be put on the left or right of the heatmap?
`row_dend_width`	Width of the row dendrogram, should be a `unit` object.
`show_row_dend`	Whether show row dendrogram?
`row_dend_gp`	Graphic parameters for the dendrogram segments. If users already provide a `dendrogram` object with edges rendered, this argument will be ignored.
`row_dend_reorder`	Apply reordering on row dendrograms. The value can be a logical value or a vector which contains weight which is used to reorder rows. The reordering is applied by `reorder.dendrogram`.
`cluster_columns`	Whether make cluster on columns? Same settings as `cluster_rows`.
`cluster_column_slices`	If columns are split into slices, whether perform clustering on the slice means?
`clustering_distance_columns`	Same setting as `clustering_distance_rows`.
`clustering_method_columns`	Method to perform hierarchical clustering, pass to `hclust`.
`column_dend_side`	Should the column dendrogram be put on the top or bottom of the heatmap?
`column_dend_height`	height of the column cluster, should be a `unit` object.
`show_column_dend`	Whether show column dendrogram?

`column_dend_gp`	Graphic parameters for dendrogram segments. Same settings as `row_dend_gp`.
`column_dend_reorder`	Apply reordering on column dendrograms. Same settings as `row_dend_reorder`.
`row_order`	Order of rows. Manually setting row order turns off clustering.
`column_order`	Order of column.
`row_labels`	Optional row labels which are put as row names in the heatmap.
`row_names_side`	Should the row names be put on the left or right of the heatmap?
`show_row_names`	Whether show row names.
`row_names_max_width`	Maximum width of row names viewport.
`row_names_gp`	Graphic parameters for row names.
`row_names_rot`	Rotation of row names.
`row_names_centered`	Should row names put centered?
`column_labels`	Optional column labels which are put as column names in the heatmap.
`column_names_side`	Should the column names be put on the top or bottom of the heatmap?
`column_names_max_height`	Maximum height of column names viewport.
`show_column_names`	Whether show column names.
`column_names_gp`	Graphic parameters for drawing text.
`column_names_rot`	Rotation of column names.
`column_names_centered`	Should column names put centered?
`top_annotation`	A `HeatmapAnnotation` object.
`bottom_annotation`	A `HeatmapAnnotation` object.
`left_annotation`	It should be specified by `rowAnnotation`.
`right_annotation`	it should be specified by `rowAnnotation`.
`km`	Apply 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 slice, hierarchical clustering is still applied with parameters above.
`split`	A vector or a data frame by which the rows are split. But if `cluster_rows` is a clustering object, `split` can be a single number indicating to split the dendrogram by `cutree`.
`row_km`	Same as `km`.
`row_km_repeats`	Number of k-means runs to get a consensus k-means clustering. Note if `row_km_repeats` is set to more than one, the final number of groups might be smaller than `row_km`, but this might means the original `row_km` is not a good choice.

`row_split`	Same as `split`.
`column_km`	K-means clustering on columns.
`column_km_repeats`	Number of k-means runs to get a consensus k-means clustering. Similar as `row_km_repeats`.
`column_split`	Split on columns. For heatmap splitting, please refer to `https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#heatmap-split` .
`gap`	Gap between row slices if the heatmap is split by rows. The value should be a `unit` object.
`row_gap`	Same as `gap`.
`column_gap`	Gap between column slices.
`show_parent_dend_line`	When heatmap is split, whether to add a dashed line to mark parent dendrogram and children dendrograms?
`width`	Width of the heatmap body.
`height`	Height of the heatmap body.
`heatmap_width`	Width of the whole heatmap (including heatmap components)
`heatmap_height`	Height of the whole heatmap (including heatmap components). Check `https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#size-of-the-heatmap` .
`show_heatmap_legend`	Whether show heatmap legend?
`heatmap_legend_param`	A list contains parameters for the heatmap legends. See `color_mapping_legend,ColorMapping-metho` for all available parameters.
`use_raster`	Whether render the heatmap body as a raster image. It helps to reduce file size when the matrix is huge. If number of rows or columns is more than 2000, it is by default turned on. Note if `cell_fun` is set, `use_raster` is enforced to be `FALSE`.
`raster_device`	Graphic device which is used to generate the raster image.
`raster_quality`	A value larger than 1.
`raster_device_param`	A list of further parameters for the selected graphic device. For raster image support, please check `https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html#heatmap-as-raster-image` .
`raster_resize_mat`	Whether resize the matrix to let the dimension of the matrix the same as the dimension of the raster image? The value can be logical. If it is `TRUE`, `mean` is used to summarize the sub matrix which corresponds to a single pixel. The value can also be a summary function, e.g. `max`.
`raster_by_magick`	Whether to use `image_resize` to scale the image.

`raster_magick_filter`
: Pass to `filter` argument of `image_resize`. A character scalar and all possible values are in `filter_types`. The default is `"Lanczos"`.

`post_fun`
: A function which will be executed after the heatmap list is drawn.

Details

The initialization function only applies parameter checking and fill values to the slots with some validation.

Following methods can be applied to the `Heatmap-class` object:

- `show,Heatmap-method`: draw a single heatmap with default parameters
- `draw,Heatmap-method`: draw a single heatmap.
- + or `%v%` append heatmaps and 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>

See Also

`https://jokergoo.github.io/ComplexHeatmap-reference/book/a-single-heatmap.html`

Examples

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

`Heatmap-class` *Class for a Single Heatmap*

Description

Class for a Single Heatmap

Details

The `Heatmap-class` is not responsible for heatmap legend and annotation legends. The `draw,Heatmap-method` method constructs a `HeatmapList-class` object which only contains one single heatmap and call `draw,HeatmapList-method` to make the 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 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

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

Heatmap3D *3D Heatmap*

Description

3D Heatmap

Usage

```
Heatmap3D(matrix,
    ...,

    bar_rel_width = 0.6,
    bar_rel_height = 0.6,
    bar_max_length = unit(1, "cm"),
    bar_angle = 60,

    row_names_side = "left",
    show_row_dend = FALSE,
    show_column_dend = FALSE)
```

Arguments

`matrix`	The input matrix. Values should be non-negative.
`...`	All pass to `Heatmap`.
`bar_rel_width`	A factor between 0 and 1.
`bar_rel_height`	A factor between 0 and 1.
`bar_max_length`	Maximal length of bars. Value should be in absolute unit.
`bar_angle`	Angle for the projection.
`row_names_side`	Row names are by default put on the left side of the heatmap.
`show_row_dend`	By default the dendrogram is not drawn.
`show_column_dend`	By default the dendrogram is not drawn.

Detals

For large matrices, the plotting might be slow.

Examples

```
m = matrix(sample(100, 36), 6)
Heatmap3D(m)
```

`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"),
    gp = gpar(col = NA),
    border = FALSE,
    gap = unit(1, "points"),

    show_annotation_name = TRUE,
    annotation_label = NULL,
    annotation_name_gp = gpar(),
    annotation_name_offset = NULL,
    annotation_name_side = ifelse(which == "column", "right", "bottom"),
    annotation_name_rot = NULL,
    annotation_name_align = FALSE,

    annotation_height = NULL,
    annotation_width = NULL,
    height = NULL,
    width = NULL,
    simple_anno_size = ht_opt$simple_anno_size,
    simple_anno_size_adjust = FALSE)
```

Arguments

`...`	Name-value pairs where the names correspond to annotation names and values can be a vector, a matrix and an annotation function. Each pair is sent to `SingleAnnotation` to contruct a single annotation.
`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 contain color mapping to `df` or simple annotations defined in `...`. See `SingleAnnotation` 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. See `color_mapping_legend,ColorMapping-me` for all possible options.
`show_legend`	Whether show annotation legends. The value can be one single value or a vector.
`which`	Are these row annotations or column annotations?
`gp`	Graphic parameters for simple annotations (with `fill` parameter ignored).
`border`	border of single annotations.
`gap`	Gap between annotations. It can be a single value or a vector of `unit` objects.
`show_annotation_name`	Whether show annotation names? For column annotation, annotation names are drawn either on the left or the right, and for row annotations, names are draw either on top or at the bottom. The value can be a vector.
`annotation_label`	Labels for the annotations. By default it is the same as individual annotation names.
`annotation_name_gp`	Graphic parameters for anntation names. Graphic paramters can be vectors.
`annotation_name_offset`	Offset to the annotation names, a `unit` object. The value can be a vector.
`annotation_name_side`	Side of the annotation names.
`annotation_name_rot`	Rotation of the annotation names. The value can be a vector.
`annotation_name_align`	Whether to align the annotation names.

`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 whole column annotations.
`width`	Width of the whole heatmap annotations.
`simple_anno_size`	Size of the simple annotation.
`simple_anno_size_adjust`	Whether also adjust the size of simple annotations when adjusting the whole heatmap annotation.

Details

For arguments `show_legend`, `border`, `annotation_name_offset`, `annotation_name_side`, `annotation_name_rot`, `show_annotation_name`, they can be set as named vectors to modify values for some of the annotations, e.g. assuming you have an annotation with name `foo`, you can specify `border = c(foo = TRUE)` in `HeatmapAnnotation`.

There are three ways to specify heatmap annotations:

1. If the annotation is simply a vector or a matrix, it can be specified like `HeatmapAnnotation(foo = 1:10)`. 2. If the annotations are already stored as a data frame, it can be specified like `HeatmapAnnotation(df = df)`. 3. For complex annotations, users can use the pre-defined annotation functions such as `anno_points`: `HeatmapAnnotation(foo = anno_points(1:10))`.

For more details and examples, please check `https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html.`

Value

A `HeatmapAnnotation-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

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

Examples

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

`HeatmapAnnotation-class` *Class for Heatmap Annotations*

Description

Class for Heatmap Annotations

Details

A complex heatmap contains a list of annotations which are represented as 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.

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

```
# There is no example
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.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is 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 annotations.

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

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

heatmap_legend_size-HeatmapList-method

Size of the Heatmap Legends

Description

Size of the Heatmap Legends

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. It is normally constructed by `Legend`.
`...`	Other arguments.

Details

Internally, all heatmap legends are packed by `packLegend` as a single `grob` object.

This function is only for internal use.

Value

A `unit` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

height.AnnotationFunction *Height of the AnnotationFunction Object*

Description

Height of the AnnotationFunction Object

Usage

```
## S3 method for class 'AnnotationFunction'
height(x, ...)
```

Arguments

`x`	The `AnnotationFunction-class` object.
`...`	Other arguments.

Details

Internally used.

Examples

```
anno = anno_points(1:10)
ComplexHeatmap:::height(anno)
anno = anno_points(1:10, which = "row")
ComplexHeatmap:::height(anno)
```

height.Heatmap *Height of the Heatmap*

Description

Height of the Heatmap

Usage

```
## S3 method for class 'Heatmap'
height(x, ...)
```

Arguments

`x`	The `HeatmapList-class` object returned by `draw,Heatmap-method`.
`...`	Other arguments.

Examples

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

`height.HeatmapAnnotation` *Height of the HeatmapAnnotation Object*

Description

Height of the HeatmapAnnotation Object

Usage

```
## S3 method for class 'HeatmapAnnotation'
height(x, ...)
```

Arguments

`x`	The `HeatmapAnnotation-class` object.
`...`	Other arguments.

Details

Internally used.

Examples

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

`height.HeatmapList` *Height of the Heatmap List*

Description

Height of the Heatmap List

Usage

```
## S3 method for class 'HeatmapList'
height(x, ...)
```

Arguments

`x`	The `HeatmapList-class` object returned by `draw,HeatmapList-method`.
`...`	Other arguments.

Examples

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

`height.Legends` *Height of the Legends*

Description

Height of the Legends

Usage

```
## S3 method for class 'Legends'
height(x, ...)
```

Arguments

`x`	The `grob` object returned by `Legend` or `packLegend`.
`...`	Other arguments.

Value

The returned unit x is always in `mm`.

Examples

```
lgd = Legend(labels = 1:10, title = "foo", legend_gp = gpar(fill = "red"))
ComplexHeatmap:::height(lgd)
```

height.SingleAnnotation

Height of the SingleAnnotation object

Description

Height of the SingleAnnotation object

Usage

```
## S3 method for class 'SingleAnnotation'
height(x, ...)
```

Arguments

`x`	The `SingleAnnotation-class` object.
`...`	Other arguments.

Details

Internally used.

Examples

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

heightAssign.AnnotationFunction

Assign the Height to the AnnotationFunction Object

Description

Assign the Height to the AnnotationFunction Object

Usage

```
## S3 replacement method for class 'AnnotationFunction'
height(x, ...) <- value
```

Arguments

`x`	The `AnnotationFunction-class` object.
`value`	A `unit` object.
`...`	Other arguments.

Details

Internally used.

Examples

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

`heightAssign.HeatmapAnnotation`
Assign the Height to the HeatmapAnnotation Object

Description

Assign the Height to the HeatmapAnnotation Object

Usage

```
## S3 replacement method for class 'HeatmapAnnotation'
height(x, ...) <- value
```

Arguments

`x`	The `HeatmapAnnotation-class` object.
`value`	A `unit` object.
`...`	Other arguments.

Details

Internally used.

Examples

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

heightAssign.SingleAnnotation

Assign the Height to the SingleAnnotation Object

Description

Assign the Height to the SingleAnnotation Object

Usage

```
## S3 replacement method for class 'SingleAnnotation'
height(x, ...) <- value
```

Arguments

`x`	The `SingleAnnotation-class` object.
`value`	A `unit` object.
`...`	Other arguments.

Details

Internally used.

Examples

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

heightDetails.annotation_axis

Height for annotation_axis Grob

Description

Height for annotation_axis Grob

Usage

```
## S3 method for class 'annotation_axis'
heightDetails(x)
```

Arguments

`x`	The `annotation_axis` grob returned by `annotation_axis_grob`.

Details

The physical height of the grob can be get by `convertWidth(grobHeight(axis_grob),"mm")`.

Examples

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

`heightDetails.legend` *Grob height for packed_legends*

Description

Grob height for packed_legends

Usage

```
## S3 method for class 'legend'
heightDetails(x)
```

Arguments

`x`	A legend object.

Examples

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

`heightDetails.legend_body` *Grob height for legend_body*

Description

Grob height for legend_body

Usage

```
## S3 method for class 'legend_body'
heightDetails(x)
```

Arguments

`x`	A legend_body object.

Examples

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

---

heightDetails.packed_legends — *Grob height for packed_legends*

Description

Grob height for packed_legends

Usage

```
## S3 method for class 'packed_legends'
heightDetails(x)
```

Arguments

`x`	A packed_legends object.

Examples

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

---

ht_global_opt — *Global Options for Heatmaps*

Description

Global Options for Heatmaps

Usage

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

Arguments

`...`	Options.
`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`	Wwitch to local mode.
`ADD`	Add new options.

Details

This function is deprecated. Please use `ht_opt` instead. However, changes by this function will also be sychronized in `ht_opt`.

Examples

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

`ht_opt` *Global Options for Heatmaps*

Description

Global Options for Heatmaps

Usage

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

Arguments

`...`	Options, see 'Details' section.
`RESET`	Reset all the option values.
`READ.ONLY`	Please ignore this argument.
`LOCAL`	Please ignore this argument.
`ADD`	Please ignore this argument.

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.

There are following parameters to control all heatmaps:

heatmap_row_names_gp set `row_names_gp` in all `Heatmap`.

heatmap_column_names_gp set `column_names_gp` in all `Heatmap`.

heatmap_row_title_gp set `row_title_gp` in all `Heatmap`.

heatmap_column_title_gp set `column_title_gp` in all `Heatmap`.

heatmap_border set `border` in all `Heatmap`.

Following parameters control the legends:

legend_title_gp set `title_gp` in all heatmap legends and annotation legends.

legend_title_position set `title_position` in all heatmap legends and annotation legends.

legend_labels_gp set `labels_gp` in all heatmap legends and annotation legends.

legend_grid_width set `grid_width` in all heatmap legends and annotation legends.

legend_grid_height set `grid_height` in all heatmap legends and annotation legends.

legend_border set `border` in all heatmap legends and annotation legends.

legend_gap Gap between legends. The value should be a vector of two units. One for gaps between vertical legends and one for the horizontal legends. If only one single unit is specified, the same gap set for the vertical and horizontal legends.

Following parameters control heatmap annotations:

annotation_border `border` in all `HeatmapAnnotation`.

simple_anno_size size for the simple annotation.

Following parameters control the space between heatmap components:

DENDROGRAM_PADDING space bewteen dendrograms and heatmap body.

DIMNAME_PADDING space between row/column names and heatmap body.

TITLE_PADDING space between row/column titles and heatmap body. The value can have length of two which corresponds to the botton and top padding.

COLUMN_ANNO_PADDING space between column annotations and heatmap body.

ROW_ANNO_PADDING space between row annotations and heatmap body.

HEATMAP_LEGEND_PADDING space between heatmap legends and heatmaps

ANNOTATION_LEGEND_PADDING space between annotation legends and heatmaps

Other parameters:

fast_hclust whether use `hclust` to speed up clustering?

show_parent_dend_line when heatmap is split, whether to add a dashed line to mark parent dendrogram and children dendrograms?

You can get or set option values by the traditional way (like `options`) or by `$` operator:

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

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

Reset to the default values by `ht_opt(RESET = TRUE)`.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
ht_opt
```

`ht_size` *Calculate the width and height of the heatmaps*

Description

Calculate the width and height of the heatmaps

Usage

```
ht_size(ht)
```

Arguments

`ht`	A `Heatmap-class` or `HeatmapList-class` object.

Value

A list of two elements: width and height.

Examples

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

`is_abs_unit` *Test Whether it is an Absolute Unit*

Description

Test Whether it is an 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 assumes `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(grobWidth(textGrob("foo")))
is_abs_unit(unit(1, "mm") + unit(1, "npc"))
```

`Legend` *Make a Single Legend*

Description

Make a Single Legend

Usage

```
Legend(at, labels = at, col_fun, name = NULL, grob = NULL,
    break_dist = NULL, nrow = NULL, ncol = 1, by_row = FALSE,
    grid_height = unit(4, "mm"),
    grid_width = unit(4, "mm"),
    gap = unit(2, "mm"), column_gap = gap, row_gap = unit(0, "mm"),
    labels_gp = gpar(fontsize = 10), labels_rot = 0,
    border = NULL, background = "#EEEEEE",
    type = "grid", graphics = NULL, 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", "leftcenter-rot", "lefttop-rot"
    title_gap = unit(2, "mm"))
```

Arguments

`at`	Breaks of the legend. The values can be either numeric or character. If it is not specified, the values of `labels` are taken as labels.
`labels`	Labels corresponding to `at`. If it is not specified, the values of `at` are taken as labels.
`col_fun`	A color mapping function which is used to make a continuous legend. Use `colorRamp2` to generate the color mapping function. If `at` is missing, the breaks recorded in the color mapping function are used for `at`.

`name`	Name of the legend, internally used.
`grob`	The legend body can be specified by a pre-constructed `grob` object.
`break_dist`	A zooming factor to control relative distance of two neighbouring break values.The length of it should be `length(at) -1` or a scalar.
`nrow`	For legend which is represented as grids, `nrow` controls number of rows of the grids if the grids are arranged into multiple rows.
`ncol`	Similar as `nrow`, `ncol` controls number of columns of the grids if the grids are arranged into multiple columns. Note at a same time only one of `nrow` and `ncol` can be specified.
`by_row`	Are the legend grids arranged by rows or by columns?
`grid_height`	The height of legend grid. It can also control the height of the continuous legend if it is horizontal.
`grid_width`	The width of legend grid. It can also control the width of the continuous legend if it is vertical.
`gap`	If legend grids are put into multiple rows or columns, this controls the gap between neighbouring rows or columns, measured as a `unit` object.
`column_gap`	The same as `gap`.
`row_gap`	Space between legend rows.
`labels_gp`	Graphic parameters for labels.
`labels_rot`	Text rotation for labels. It should only be used for horizontal continuous legend.
`border`	Color of legend grid borders. It also works for the ticks in the continuous legend.
`background`	Background colors for the grids. It is used when points and lines are the legend graphics.
`type`	Type of legends. The value can be one of `grid`, `points`, `lines` and `boxplot`.
`graphics`	Self-defined graphics for legends. The value should be a list of functions. Each function should accept four argumets: `x` and `y`: positions of the legend grid (center point), `w` and `h`: width and height of the legend grid.
`legend_gp`	Graphic parameters for the legend grids. You should control the filled color of the legend grids by `gpar(fill = ...)`.
`pch`	Type of points if points are used as legend. Note you can use single-letter as pch, e.g. `pch = 'A'`. There are three additional integers that are valid for `pch`: 26 and 27 for single diagonal lines and 28 for double diagonal lines.
`size`	Size of points.
`legend_height`	Height of the whole legend body. It is only used for vertical continous legend.
`legend_width`	Width of the whole legend body. It is only used for horizontal continous legend.
`direction`	Direction of the legend, vertical or horizontal?
`title`	Title of the legend.
`title_gp`	Graphic parameters of the title.
`title_position`	Position of title relative to the legend. `topleft`, `topcenter`, `leftcenter-rot` and `lefttop-rot` are only for vertical legend and `leftcenter`, `lefttop` are only for horizontal legend.
`title_gap`	Gap between title and the legend body.

Details

Most of the argument can also be set in `heatmap_legend_param` argument in `Heatmap` or `annotation_legend_param` argument in `HeatmapAnnotation` to configure legend styles for heatmap and annotations.

Value

A `Legends-class` object.

See Also

`packLegend` packs multiple legends into one `Legends-class` object.

See examples of configuring legends: `https://jokergoo.github.io/ComplexHeatmap-reference/book/legends.html`

Examples

```
lgd = Legend(labels = month.name[1:6], title = "foo", legend_gp = gpar(fill = 1:6))
draw(lgd, test = "add labels and title")

require(circlize)
col_fun = colorRamp2(c(0, 0.5, 1), c("blue", "white", "red"))
lgd = Legend(col_fun = col_fun, title = "foo")
draw(lgd, test = "only col_fun")

col_fun = colorRamp2(c(0, 0.5, 1), c("blue", "white", "red"))
lgd = Legend(col_fun = col_fun, title = "foo", at = c(0, 0.1, 0.15, 0.5, 0.9, 0.95, 1))
draw(lgd, test = "unequal interval breaks")
```

`Legends` *Constructor method for Legends class*

Description

Constructor method for Legends class

Usage

```
Legends(...)
```

Arguments

`...`	arguments.

Details

There is no public constructor method for the `Legends-class`.

Value

No value is returned.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

`Legends-class` *The Class for Legends*

Description

The Class for Legends

Details

This is a very simple class for legends that it only has one slot which is the real `grob` of the legends.

Construct a single legend by `Legend` and a group of legends by `packLegend`.

Examples

```
lgd = Legend(at = 1:4)
lgd
lgd@grob
```

`length.HeatmapAnnotation`
Number of Annotations

Description

Number of Annotations

Usage

```
## S3 method for class 'HeatmapAnnotation'
length(x)
```

Arguments

`x`	A `HeatmapAnnotation-class` object.

Examples

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

`length.HeatmapList` *Length of the HeatmapList object*

Description

Length of the HeatmapList object

Usage

```
## S3 method for class 'HeatmapList'
length(x)
```

Arguments

`x`	A `HeatmapList-class` object

Examples

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

`list_components` *List All Heatmap Components*

Description

List All Heatmap Components

Usage

```
list_components(pattern = NULL)
```

Arguments

`pattern`	A regular expression.

Value

A vector of viewport names.

Examples

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

`list_to_matrix` *Convert a List of Sets to a Binary Matrix*

Description

Convert a List of Sets to a Binary Matrix

Usage

```
list_to_matrix(lt, universal_set = NULL)
```

Arguments

`lt`	A list of vectors.
`universal_set`	The universal set.

Details

It converts the list which have m sets to a binary matrix with n rows and m columns where n is the size of universal set.

Examples

```
set.seed(123)
lt = list(a = sample(letters, 5),
          b = sample(letters, 10),
          c = sample(letters, 15))
list_to_matrix(lt)
list_to_matrix(lt, universal_set = letters)
```

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_list`, `column_order_list`, `column_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

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

make_comb_mat	*Make a Combination Matrix for UpSet Plot*

Description

Make a Combination Matrix for UpSet Plot

Usage

```
make_comb_mat(..., mode = c("distinct", "intersect", "union"),
    top_n_sets = Inf, min_set_size = -Inf,
    universal_set = NULL, complement_size = NULL,
    value_fun = NULL, set_on_rows = TRUE)
```

Arguments

`...`	The input sets. If it is represented as a single variable, it should be a matrix/data frame or a list. If it is multiple variables, it should be name-value pairs, see Input section for explanation.
`mode`	The mode for forming the combination set, see Mode section.
`top_n_sets`	Number of sets with largest size.
`min_set_size`	Ths minimal set size that is used for generating the combination matrix.
`universal_set`	The universal set. If it is set, the size of the complement set of all sets is also calculated. It if is specified, `complement_size` is ignored.
`complement_size`	The size for the complement of all sets. If it is specified, the combination set name will be like "00...".
`value_fun`	For each combination set, how to calculate the size? If it is a scalar set, the length of the vector is the size of the set, while if it is a region-based set, (i.e. `GRanges` or `IRanges` object), the sum of widths of regions in the set is calculated as the size of the set.
`set_on_rows`	Used internally.

Value

A matrix also in a class of `comb_mat`.

Following functions can be applied to it: `set_name`, `comb_name`, `set_size`, `comb_size`, `comb_degree`, `extract_comb` and `t.comb_mat`.

Input

To represent multiple sets, the variable can be represented as:

1. A list of sets where each set is a vector, e.g.:

```
list(set1 = c("a", "b", "c"),
     set2 = c("b", "c", "d", "e"),
     ...)
```

2. A binary matrix/data frame where rows are elements and columns are sets, e.g.:

```
  a b c
h 1 1 1
t 1 0 1
j 1 0 0
u 1 0 1
w 1 0 0
...
```

If the variable is a data frame, the binary columns (only contain 0 and 1) and the logical columns are only kept.

The set can be genomic regions, then it can only be represented as a list of `GRanges` objects.

Mode

E.g. for three sets (A, B, C), the UpSet approach splits the combination of selecting elements in the set or not in the set and calculates the sizes of the combination sets. For three sets, all possible combinations are:

```
A B C
1 1 1
1 1 0
1 0 1
0 1 1
1 0 0
0 1 0
0 0 1
```

A value of 1 means to select that set and 0 means not to select that set. E.g., "1 1 0" means to select set A, B while not set C. Note there is no "0 0 0", because the background size is not of interest here. With the code of selecting and not selecting the sets, next we need to define how to calculate the size of that combination set. There are three modes:

1. `distinct` mode: 1 means in that set and 0 means not in that set, then "1 1 0" means a set of elements also in set A and B, while not in C (i.e. `setdiff(intersect(A,B),C)`). Under this mode, the seven combination sets are the seven partitions in the Venn diagram and they are mutually exclusive.

2. `intersect` mode: 1 means in that set and 0 is not taken into account, then, "1 1 0" means a set of elements in set A and B, and they can also in C or not in C (i.e. `intersect(A,B)`). Under this mode, the seven combination sets can overlap.

3. `union` mode: 1 means in that set and 0 is not taken into account. When there are multiple 1, the relationship is OR. Then, "1 1 0" means a set of elements in set A or B, and they can also in C or not in C (i.e. `union(A,B)`). Under this mode, the seven combination sets can overlap.

Examples

```
set.seed(123)
lt = list(a = sample(letters, 10),
          b = sample(letters, 15),
          c = sample(letters, 20))
m = make_comb_mat(lt)

mat = list_to_matrix(lt)
mat
m = make_comb_mat(mat)

## Not run:
require(circlize)
require(GenomicRanges)
lt = lapply(1:4, function(i) generateRandomBed())
lt = lapply(lt, function(df) GRanges(seqnames = df[, 1],
    ranges = IRanges(df[, 2], df[, 3])))
names(lt) = letters[1:4]
m = make_comb_mat(lt)

## End(Not run)
```

---

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,Heatmap-method`, `Heatmap-class` class method
- `make_layout,HeatmapList-method`, `HeatmapList-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 component. Also how to make graphics for heatmap components will be recorded by saving as functions.

Whether to apply row clustering or column clustering affects the layout, so clustering should be applied first by `prepare,Heatmap-method` 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

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

make_layout-HeatmapList-method
Make Layout for the Heatmap List

Description

Make Layout for the Heatmap List

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(),
    align_heatmap_legend = NULL,
    align_annotation_legend = NULL,
    legend_grouping = c("adjusted", "original"),

    ht_gap = unit(2, "mm"),

    main_heatmap = which(sapply(object@ht_list, inherits, "Heatmap"))[1],
    padding = GLOBAL_PADDING,

    auto_adjust = TRUE,
    row_dend_side = c("original", "left", "right"),
    row_sub_title_side = c("original", "left", "right"),
    column_dend_side = c("original", "top", "bottom"),
    column_sub_title_side = c("original", "top", "bottom"),

    row_gap = NULL,
    cluster_rows = NULL,
    cluster_row_slices = 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,
    row_km = NULL,
    row_km_repeats = NULL,
    row_split = NULL,
    height = NULL,
    heatmap_height = NULL,

    column_gap = NULL,
    cluster_columns = NULL,
    cluster_column_slices = NULL,
    clustering_distance_columns = NULL,
    clustering_method_columns = NULL,
    column_dend_width = NULL,
    show_column_dend = NULL,
    column_dend_reorder = NULL,
    column_dend_gp = NULL,
    column_order = NULL,
    column_km = NULL,
    column_km_repeats = NULL,
    column_split = NULL,
    width = NULL,
    heatmap_width = NULL,

    use_raster = NULL,
    raster_device = NULL,
    raster_quality = NULL,
    raster_device_param = NULL,
    raster_resize = 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 list?
`row_title_gp`	Graphic parameters for the row title.
`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 the column title.
`heatmap_legend_side`	Side of the heatmap legends.

`merge_legends`	Whether to put heatmap legends and annotation legends together. By default they are put in different viewports.
`show_heatmap_legend`	Whether show heatmap legends.
`heatmap_legend_list`	A list of self-defined legends, should be wrapped into a list of `grob` objects. Normally they are constructed by `Legend`.
`annotation_legend_side`	Side of annotation legends.
`show_annotation_legend`	Whether show annotation legends.
`annotation_legend_list`	A list of self-defined legends, should be wrapped into a list of `grob` objects. Normally they are constructed by `Legend`.
`align_heatmap_legend`	How to align the legends to heatmap. Possible values are "heatmap_center", "heatmap_top" and "global_center". If the value is `NULL`, it automatically picks the proper value from the three options.
`align_annotation_legend`	How to align the legends to heatmap. Possible values are "heatmap_center", "heatmap_top" and "global_center".
`legend_grouping`	How the legends are grouped. Values should be "adjusted" or "original".
`ht_gap`	Gap between heatmaps, should be a `unit` object. It can be a vector of length 1 or the number of heamtaps/annotations.
`main_heatmap`	Name or index for the main heatmap.
`padding`	Padding of the whole plot. The four values correspond to the bottom, left, top and right paddings.
`auto_adjust`	whether apply automatic adjustment? The auto-adjustment includes turning off dendrograms, titles and row/columns for non-main heatmaps.
`row_dend_side`	If auto-adjustment is on, to put the row dendrograms of the main heatmap to the most left side of the heatmap list or the most right side?
`row_sub_title_side`	There can be sub titles generated by the splitting of heatmaps. Similar setting as `row_dend_side`.
`column_dend_side`	Similar setting as `row_dend_side`.
`column_sub_title_side`	Similar setting as `row_sub_title_side`.
`row_gap`	Overwrite the corresponding setting in the main heatmap.
`cluster_rows`	Overwrite the corresponding setting in the main heatmap.
`cluster_row_slices`	Overwrite the corresponding setting in the main heatmap.

`clustering_distance_rows`	Overwrite the corresponding setting in the main heatmap.
`clustering_method_rows`	Overwrite the corresponding setting in the main heatmap.same setting as in `Heatmap`, if it is specified, `clustering_method_rows` in main heatmap is ignored.
`row_dend_width`	Overwrite the corresponding setting in the main heatmap.
`show_row_dend`	same Overwrite the corresponding setting in the main heatmap.
`row_dend_reorder`	Overwrite the corresponding setting in the main heatmap.
`row_dend_gp`	Overwrite the corresponding setting in the main heatmap.
`row_order`	Overwrite the corresponding setting in the main heatmap.
`row_km`	Overwrite the corresponding setting in the main heatmap.
`row_km_repeats`	Overwrite the corresponding setting in the main heatmap.
`row_split`	Overwrite the corresponding setting in the main heatmap.
`height`	Overwrite the corresponding setting in the main heatmap.
`heatmap_height`	Overwrite the corresponding setting in the main heatmap.
`column_gap`	Overwrite the corresponding setting in the main heatmap.
`cluster_columns`	Overwrite the corresponding setting in the main heatmap.
`cluster_column_slices`	Overwrite the corresponding setting in the main heatmap.
`clustering_distance_columns`	Overwrite the corresponding setting in the main heatmap.
`clustering_method_columns`	Overwrite the corresponding setting in the main heatmap.
`column_dend_width`	column Overwrite the corresponding setting in the main heatmap.
`show_column_dend`	Overwrite the corresponding setting in the main heatmap.
`column_dend_reorder`	Overwrite the corresponding setting in the main heatmap.
`column_dend_gp`	Overwrite the corresponding setting in the main heatmap.
`column_order`	Overwrite the corresponding setting in the main heatmap.
`column_km`	Overwrite the corresponding setting in the main heatmap.
`column_km_repeats`	Overwrite the corresponding setting in the main heatmap.
`column_split`	Overwrite the corresponding setting in the main heatmap.
`width`	Overwrite the corresponding setting in the main heatmap.
`heatmap_width`	Overwrite the corresponding setting in the main heatmap.
`use_raster`	Overwrite the corresponding setting in every heatmap.

`raster_device`	Overwrite the corresponding setting in every heatmap.
`raster_quality`	Overwrite the corresponding setting in every heatmap.
`raster_device_param`	Overwrite the corresponding setting in every heatmap.
`raster_resize`	Overwrite the corresponding setting in every heatmap.

Details

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

This function is only for internal use.

Value

A `HeatmapList-class` object in which settings for all heatmap are adjusted.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
# There is no example
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

```
# There is no example
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 ColorMapping-class object.
x	Input values.

Details

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

This function provides a uniform way for discrete and continuous color mapping.

Value

A vector of colors.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
cm = ColorMapping(colors = c("A" = "red", "B" = "black"))
map_to_colors(cm, sample(c("A", "B"), 10, replace = TRUE))
require(circlize)
col_fun = colorRamp2(c(0, 1), c("white", "red"))
cm = ColorMapping(col_fun = col_fun)
map_to_colors(cm, runif(10))
```

---

max_text_height *Maximum Height of Text*

Description

Maximum Height of Text

Usage

```
max_text_height(text, gp = gpar(), rot = 0)
```

Arguments

text	A vector of text.
gp	Graphic parameters for text.
rot	Rotation of the text, scalar.

Details

It simply calculates maximum height of a list of `textGrob` objects.

Note it ignores the text rotation.

Value

A `unit` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`max_text_width` calculates the maximum width of a text vector.

Examples

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

max_text_width *Maximum Width of Text*

Description

Maximum Width of Text

Usage

```
max_text_width(text, gp = gpar(), rot = 0)
```

Arguments

text	A vector of text.
gp	Graphic parameters for text.
rot	Rotation of the text, scalar.

Details

It simply calculates maximum width of a list of textGrob objects.

Note it ignores the text rotation.

Value

A unit object which is in "mm".

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

max_text_height calculates the maximum height of a text vector.

Examples

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

merge_dendrogram *Merge Dendrograms*

Description

Merge Dendrograms

Usage

```
merge_dendrogram(x, y, only_parent = FALSE, ...)
```

Arguments

x	The parent dendrogram.
y	The children dendrograms. They are connected to the leaves of the parent dendrogram. So the length of y should be as same as the number of leaves of the parent dendrogram.
only_parent	Whether only returns the parent dendrogram where the height and node positions have been adjusted by children dendrograms.
...	Other arguments.

Details

Do not retrieve the order of the merged dendrogram. It is not reliable.

Examples

```
m1 = matrix(rnorm(100), nr = 10)
m2 = matrix(rnorm(80), nr = 8)
m3 = matrix(rnorm(50), nr = 5)
dend1 = as.dendrogram(hclust(dist(m1)))
dend2 = as.dendrogram(hclust(dist(m2)))
dend3 = as.dendrogram(hclust(dist(m3)))
dend_p = as.dendrogram(hclust(dist(rbind(colMeans(m1), colMeans(m2), colMeans(m3)))))
dend_m = merge_dendrogram(dend_p, list(dend1, dend2, dend3))
grid.dendrogram(dend_m, test = TRUE)

dend_m = merge_dendrogram(dend_p, list(dend1, dend2, dend3), only_parent = TRUE)
grid.dendrogram(dend_m, test = TRUE)

require(dendextend)
dend1 = color_branches(dend1, k = 1, col = "red")
dend2 = color_branches(dend2, k = 1, col = "blue")
dend3 = color_branches(dend3, k = 1, col = "green")
dend_p = color_branches(dend_p, k = 1, col = "orange")
dend_m = merge_dendrogram(dend_p, list(dend1, dend2, dend3))
grid.dendrogram(dend_m, test = TRUE)
```

`names.HeatmapAnnotation` *Annotation Names*

Description

Annotation Names

Usage

```
## S3 method for class 'HeatmapAnnotation'
names(x)
```

Arguments

x	A `HeatmapAnnotation-class` object.

Examples

```
ha = HeatmapAnnotation(foo = 1:10, bar = anno_points(10:1))
names(ha)
```

`names.HeatmapList` *Names of the heatmaps/annotations*

Description

Names of the heatmaps/annotations

Usage

```
## S3 method for class 'HeatmapList'
names(x)
```

Arguments

x	A `HeatmapList-class` object

Examples

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

namesAssign.HeatmapAnnotation

Assign Annotation Names

Description

Assign Annotation Names

Usage

```
## S3 replacement method for class 'HeatmapAnnotation'
names(x) <- value
```

Arguments

`x`	A `HeatmapAnnotation-class` object.
`value`	A vector of new names.

Examples

```
ha = HeatmapAnnotation(foo = 1:10, bar = anno_points(10:1))
names(ha) = c("A", "B")
names(ha)
```

ncol.Heatmap

Number of Columns in the Heatmap

Description

Number of Columns in the Heatmap

Usage

```
## S3 method for class 'Heatmap'
ncol(x)
```

Arguments

`x`	A `Heatmap-class` object.

Examples

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

nobs.AnnotationFunction *Number of Observations*

Description

Number of Observations

Usage

```
## S3 method for class 'AnnotationFunction'
nobs(object, ...)
```

Arguments

`object`	The `AnnotationFunction-class` object.
`...`	Other arguments.

Details

returns `NA`.

Examples

```
anno = anno_points(1:10)
nobs(anno)
```

nobs.HeatmapAnnotation *Number of Observations*

Description

Number of Observations

Usage

```
## S3 method for class 'HeatmapAnnotation'
nobs(object, ...)
```

Arguments

`object`	The `HeatmapAnnotation-class` object.
`...`	other arguments.

Value

If there is no nobs information for any of its SingleAnnotation-class object, it returns NA.

Examples

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

nobs.SingleAnnotation *Number of Observations*

Description

Number of Observations

Usage

```
## S3 method for class 'SingleAnnotation'
nobs(object, ...)
```

Arguments

object	The SingleAnnotation-class object.
...	Other arguments.

Details

It returns the n slot of the annotaton function. If it does not exist, it returns NA.

Examples

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

`normalize_comb_mat` *Normalize a list of combination matrice*

Description

Normalize a list of combination matrice

Usage

```
normalize_comb_mat(..., full_comb_sets = FALSE, complement_set = FALSE)
```

Arguments

`...`	Combination matrices.
`full_comb_sets`	Whether the combination matrices contain the full sets of combination sets?
`complement_set`	Whether the combination matrices also contain the complement set?

Details

It normalizes a list of combination matrice to make them have same number and order of sets and combination sets.

The sets (by `set_name`) from all combination matrice should be the same.

Examples

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

`normalize_genomic_signals_to_bins` *Overlap genomic signals to the genomic bins*

Description

Overlap genomic signals to the genomic bins

Usage

```
normalize_genomic_signals_to_bins(gr, value, value_column = NULL, method = "weighted",
    empty_value = NA, window = GHEATMAP_ENV$chr_window)
```

Arguments

`gr`	A `GRanges` object.
`value`	The corresponding signals corresponding to `gr`.
`value_column`	If `value` is not set and the values are in the meta-columns in `gr`, you can specify the column indices for these value columns, better to use name indices.
`method`	One of "weighted", "w0" and "absolute". For the three different methods, please refer to `https://bioconductor.org/packages/release/bioc/vignettes/EnrichedHeatmap/inst/doc/EnrichedHeatmap.html#toc_7` .
`empty_value`	The value for the bins where no signal is overlapped.
`window`	The genomic bins generated from `bin_genome`.

Details

The genomic bins should be generated by `bin_genome` in advance. The genomic bins are saved internally, so that multiple uses of `bin_genome` ensure they all return the matrices with the same rows.

It supports following values.

- When neither `value` nor `value_column` is set, it simply overlap `gr` to the genomic bins and returns a one-column logical matrix which represents whether the current genomic bin overlaps to any signal.
- When the signals are numeric, `value` can be a numeric vector or a matrix, or `value_column` can contain multiple columns. The function returns a numeric matrix where the values are properly averaged depending on what `method` was used.
- When the signals are character, `value` can only be a vector or `value_column` can only contain one single column. The function returns a one-column character matrix.

Value

A matrix with the same row as the genomic bins.

Examples

```
## Not run:
require(circlize)
require(GenomicRanges)

chr_window = bin_genome("hg19")

#### the first is a numeric matrix #######
bed1 = generateRandomBed(nr = 1000, nc = 10)
gr1 = GRanges(seqnames = bed1[, 1], ranges = IRanges(bed1[, 2], bed1[, 3]))

num_mat = normalize_genomic_signals_to_bins(gr1, bed1[, -(1:3)])

#### the second is a character matrix ######
bed_list = lapply(1:10, function(i) {
    generateRandomBed(nr = 1000, nc = 1,
        fun = function(n) sample(c("gain", "loss"), n, replace = TRUE))
})
char_mat = NULL
for(i in 1:10) {
    bed = bed_list[[i]]
    bed = bed[sample(nrow(bed), 20), , drop = FALSE]
    gr_cnv = GRanges(seqnames = bed[, 1], ranges = IRanges(bed[, 2], bed[, 3]))

    char_mat = cbind(char_mat, normalize_genomic_signals_to_bins(gr_cnv, bed[, 4]))
}

#### two numeric columns ##########
bed2 = generateRandomBed(nr = 100, nc = 2)
gr2 = GRanges(seqnames = bed2[, 1], ranges = IRanges(bed2[, 2], bed2[, 3]))

v = normalize_genomic_signals_to_bins(gr2, bed2[, 4:5])

##### a list of genes need to be highlighted
bed3 = generateRandomBed(nr = 40, nc = 0)
gr3 = GRanges(seqnames = bed3[, 1], ranges = IRanges(bed3[, 2], bed3[, 2]))
gr3$gene = paste0("gene_", 1:length(gr3))

mtch = as.matrix(findOverlaps(chr_window, gr3))
at = mtch[, 1]
labels = mcols(gr3)[mtch[, 2], 1]

##### order of the chromosomes ########
chr = as.vector(seqnames(chr_window))
chr_level = paste0("chr", c(1:22, "X", "Y"))
chr = factor(chr, levels = chr_level)

#### make the heatmap #######
subgroup = rep(c("A", "B"), each = 5)

ht_opt$TITLE_PADDING = unit(c(4, 4), "points")
ht_list = Heatmap(num_mat, name = "mat", col = colorRamp2(c(-1, 0, 1), c("green", "white", "red")),
    row_split = chr, cluster_rows = FALSE, show_column_dend = FALSE,
    column_split = subgroup, cluster_column_slices = FALSE,
    column_title = "numeric matrix",
   top_annotation = HeatmapAnnotation(subgroup = subgroup, annotation_name_side = "left"),
    row_title_rot = 0, row_title_gp = gpar(fontsize = 10), border = TRUE,
    row_gap = unit(0, "points")) +
Heatmap(char_mat, name = "CNV", col = c("gain" = "red", "loss" = "blue"),
    border = TRUE, column_title = "character matrix") +
rowAnnotation(label = anno_mark(at = at, labels = labels)) +
rowAnnotation(pt = anno_points(v, gp = gpar(col = 4:5), pch = c(1, 16)),
    width = unit(2, "cm")) +
rowAnnotation(bar = anno_barplot(v[, 1], gp = gpar(col = ifelse(v[ ,1] > 0, 2, 3))),
    width = unit(2, "cm"))
draw(ht_list, merge_legend = TRUE)

##### or horizontally ###
```

```
ht_list = Heatmap(t(num_mat), name = "mat", col = colorRamp2(c(-1, 0, 1), c("green", "white", "red")),
    column_split = chr, cluster_columns = FALSE, show_row_dend = FALSE,
    row_split = subgroup, cluster_row_slices = FALSE,
    row_title = "numeric matrix",
    left_annotation = rowAnnotation(subgroup = subgroup, show_annotation_name = FALSE,
        annotation_legend_param = list(
         subgroup = list(direction = "horizontal", title_position = "lefttop", nrow = 1))),
    column_title_gp = gpar(fontsize = 10), border = TRUE,
    column_gap = unit(0, "points"),
   column_title = ifelse(seq_along(chr_level) %% 2 == 0, paste0("\n", chr_level), paste0(chr_level, "\n")),
   heatmap_legend_param = list(direction = "horizontal", title_position = "lefttop")) %v%
Heatmap(t(char_mat), name = "CNV", col = c("gain" = "red", "loss" = "blue"),
    border = TRUE, row_title = "character matrix",
   heatmap_legend_param = list(direction = "horizontal", title_position = "lefttop", nrow = 1)) %v%
HeatmapAnnotation(label = anno_mark(at = at, labels = labels, side = "bottom")) %v%
HeatmapAnnotation(pt = anno_points(v, gp = gpar(col = 4:5), pch = c(1, 16)),
    annotation_name_side = "left", height = unit(2, "cm")) %v%
HeatmapAnnotation(bar = anno_barplot(v[, 1], gp = gpar(col = ifelse(v[ ,1] > 0, 2, 3))),
    annotation_name_side = "left", height = unit(2, "cm"))
draw(ht_list, heatmap_legend_side = "bottom", merge_legend = TRUE)

## End(Not run)
```

nrow.Heatmap *Number of Rows in the Heatmap*

Description

Number of Rows in the Heatmap

Usage

```
## S3 method for class 'Heatmap'
nrow(x)
```

Arguments

x	A Heatmap-class object.

Examples

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

oncoPrint *Make oncoPrint*

Description

Make oncoPrint

Usage

```
oncoPrint(mat, name,
    get_type = default_get_type,
    alter_fun,
    alter_fun_is_vectorized = NULL,
    col = NULL,

    top_annotation = HeatmapAnnotation(cbar = anno_oncoprint_barplot()),
    right_annotation = rowAnnotation(rbar = anno_oncoprint_barplot()),
    left_annotation = NULL,
    bottom_annotation = NULL,

    show_pct = TRUE,
    pct_gp = gpar(fontsize = 10),
    pct_digits = 0,
    pct_side = "left",

    row_labels = NULL,
    show_row_names = TRUE,
    row_names_side = "right",
    row_names_gp = pct_gp,
    row_split = NULL,

    column_labels = NULL,
    column_names_gp = gpar(fontsize = 10),
    column_split = NULL,

    row_order = NULL,
    column_order = NULL,
    cluster_rows = FALSE,
    cluster_columns = FALSE,

    remove_empty_columns = FALSE,
    remove_empty_rows = FALSE,
    show_column_names = FALSE,
    heatmap_legend_param = NULL,
    ...)
```

Arguments

`mat`	The value should be a character matrix which encodes mulitple alterations or a list of matrices for which every matrix contains binary value representing whether the alteration is present or absent. When the value is a list, the names of the list represent alteration types. You can use `unify_mat_list` to make all matrix having same row names and column names.
`name`	Name of the oncoPrint. Not necessary to specify.
`get_type`	If different alterations are encoded in the matrix as complex strings, this self-defined function determines how to extract them. It only works when `mat` is a matrix. The default value is `default_get_type`.
`alter_fun`	A single function or a list of functions which defines how to add graphics for different alterations. You can use `alter_graphic` to automatically generate for rectangles and points.
`alter_fun_is_vectorized`	Whether `alter_fun` is implemented vectorized. Internally the function will guess.
`col`	A vector of color for which names correspond to alteration types.
`top_annotation`	Annotation put on top of the oncoPrint. By default it is barplot which shows the number of genes with a certain alteration in each sample.
`right_annotation`	Annotation put on the right of the oncoPrint. By default it is barplot which shows the number of samples with a certain alteration in each gene.
`left_annotation`	Annotation put on the left of the oncoPrint.
`bottom_annotation`	Annotation put at the bottom of the oncoPrint.
`show_pct`	whether show percent values on the left of the oncoprint?
`pct_gp`	Graphic paramters for percent values
`pct_digits`	Digits for the percent values.
`pct_side`	Side of the percent values to the oncoPrint. This argument is currently disabled.
`row_labels`	Labels as the row names of the oncoPrint.
`show_row_names`	Whether show row names?
`row_names_side`	Side of the row names to the oncoPrint. This argument is currently disabled.
`row_names_gp`	Graphic parameters for the row names.
`row_split`	Pass to `Heatmap`.
`column_labels`	Pass to `Heatmap`.
`column_names_gp`	Pass to `Heatmap`.
`column_split`	Pass to `Heatmap`.
`row_order`	Order of rows. By default rows are sorted by the number of occurence of the alterations.

`cluster_rows`	If it is set, it must be a dendrogram/hclust object.
`cluster_columns`	If it is set, it must be a dendrogram/hclust object.
`column_order`	Order of columns. By default the columns are sorted to show the mutual exclusivity of alterations.
`remove_empty_columns`	If there is no alteration in some samples, whether remove them on the oncoPrint?
`remove_empty_rows`	If there is no alteration in some samples, whether remove them on the oncoPrint?
`show_column_names`	Whether show column names?
`heatmap_legend_param`	pass to `Heatmap`.
`...`	Pass to `Heatmap`.

Details

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

`https://jokergoo.github.io/ComplexHeatmap-reference/book/oncoprint.html` gives details for configuring a oncoPrint.

Value

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

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

---

order.comb_mat *Order of the Combination Sets*

Description

Order of the Combination Sets

Usage

```
order.comb_mat(m, decreasing = TRUE, on = "comb_set")
```

Arguments

`m`	A combination matrix returned by `make_comb_mat`.
`on`	On sets or on combination sets?
`decreasing`	Whether the ordering is applied decreasingly.

Details

It first sorts by the degree of the combination sets then by the combination matrix.

Examples

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

`packLegend` *Pack Legends*

Description

Pack Legends

Usage

```
packLegend(..., gap = unit(4, "mm"), row_gap = unit(4, "mm"), column_gap = unit(4, "mm"),
    direction = c("vertical", "horizontal"),
    max_width = NULL, max_height = NULL, list = NULL)
```

Arguments

`...`	A list of objects returned by `Legend`.
`gap`	Gap between two neighbouring legends. The value is a `unit` object with length of one. It is the same as `row_gap` if the direction if vertial and the same as `column_gap` if the direction is horizontal.
`row_gap`	Horizontal gaps between legends.
`column_gap`	Vertical gaps between legends.
`direction`	The direction to arrange legends.
`max_width`	The maximal width of the total packed legends. It only works for horizontal arrangement. If the total width of the legends exceeds it, the legends will be arranged into multiple rows.
`max_height`	Similar as `max_width`, but for the vertical arrangment of legends.
`list`	The list of legends can be specified as a list.

Value

A Legends-class object.

See Also

https://jokergoo.github.io/ComplexHeatmap-reference/book/legends.html#a-list-of-legends

Examples

```
require(circlize)
col_fun = colorRamp2(c(0, 0.5, 1), c("blue", "white", "red"))
lgd1 = Legend(at = 1:6, legend_gp = gpar(fill = 1:6), title = "legend1")
lgd2 = Legend(col_fun = col_fun, title = "legend2", at = c(0, 0.25, 0.5, 0.75, 1))
pd = packLegend(lgd1, lgd2)
draw(pd, test = "two legends")
pd = packLegend(lgd1, lgd2, direction = "horizontal")
draw(pd, test = "two legends packed horizontally")
```

---

pheatmap *Translate pheatmap::pheatmap to ComplexHeatmap::Heatmap*

Description

Translate pheatmap::pheatmap to ComplexHeatmap::Heatmap

Usage

```
pheatmap(mat,
    color = colorRampPalette(rev(brewer.pal(n = 7, name = "RdYlBu")))(100),
    kmeans_k = NA,
    breaks = NA,
    border_color = ifelse(nrow(mat) < 100 & ncol(mat) < 100, "grey60", NA),
    cellwidth = NA,
    cellheight = NA,
    scale = "none",
    cluster_rows = TRUE,
    cluster_cols = TRUE,
    clustering_distance_rows = "euclidean",
    clustering_distance_cols = "euclidean",
    clustering_method = "complete",
    clustering_callback = NA,
    cutree_rows = NA,
    cutree_cols = NA,
   treeheight_row = ifelse(class(cluster_rows) == "hclust" || cluster_rows, 50, 0),
   treeheight_col = ifelse(class(cluster_cols) == "hclust" || cluster_cols, 50, 0),
    legend = TRUE,
    legend_breaks = NA,
    legend_labels = NA,
    annotation_row = NA,
    annotation_col = NA,
    annotation = NA,
    annotation_colors = NA,
    annotation_legend = TRUE,
    annotation_names_row = TRUE,
    annotation_names_col = TRUE,
    drop_levels = TRUE,
    show_rownames = TRUE,
    show_colnames = TRUE,
    main = NA,
    fontsize = 10,
    fontsize_row = fontsize,
    fontsize_col = fontsize,
    angle_col = c("270", "0", "45", "90", "315"),
    display_numbers = FALSE,
    number_format = "%.2f",
    number_color = "grey30",
    fontsize_number = 0.8 * fontsize,
    gaps_row = NULL,
    gaps_col = NULL,
    labels_row = NULL,
    labels_col = NULL,
    filename = NA,
    width = NA,
    height = NA,
    silent = FALSE,
    na_col = "#DDDDDD",
    name = NULL,

    # argument specific for Heatmap()
    heatmap_legend_param = list(),
    ...,
    run_draw = FALSE)
```

Arguments

`mat`	The input matrix.
`color`	The same as in pheatmap. Here you don't necessarily need to generate a long color vector. The discrete colors sent to `colorRampPalette` are also OK here. E.g. `colorRampPalette(rev(brewer.pal(n = 7,name = "RdYlBu")))(100)` can be simply replaced as `rev(brewer.pal(n = 7,name = "RdYlBu"))`.
`kmeans_k`	The same as in pheatmap.
`breaks`	The same as in pheatmap.
`border_color`	The same as in pheatmap.
`cellwidth`	The same as in pheatmap.

`cellheight`	The same as in pheatmap.
`scale`	The same as in pheatmap.
`cluster_rows`	The same as in pheatmap.
`cluster_cols`	The same as in pheatmap.
`clustering_distance_rows`	The same as in pheatmap.
`clustering_distance_cols`	The same as in pheatmap.
`clustering_method`	The same as in pheatmap.
`clustering_callback`	The same as in pheatmap.
`cutree_rows`	The same as in pheatmap.
`cutree_cols`	The same as in pheatmap.
`treeheight_row`	The same as in pheatmap.
`treeheight_col`	The same as in pheatmap.
`legend`	The same as in pheatmap.
`legend_breaks`	The same as in pheatmap.
`legend_labels`	The same as in pheatmap.
`annotation_row`	The same as in pheatmap.
`annotation_col`	The same as in pheatmap.
`annotation`	The same as in pheatmap.
`annotation_colors`	The same as in pheatmap.
`annotation_legend`	The same as in pheatmap.
`annotation_names_row`	The same as in pheatmap.
`annotation_names_col`	The same as in pheatmap.
`drop_levels`	Enforced to be `TRUE`.
`show_rownames`	The same as in pheatmap.
`show_colnames`	The same as in pheatmap.
`main`	The same as in pheatmap.
`fontsize`	The same as in pheatmap.
`fontsize_row`	The same as in pheatmap.
`fontsize_col`	The same as in pheatmap.
`angle_col`	The same as in pheatmap.
`display_numbers`	The same as in pheatmap.

`number_format`	The same as in `pheatmap`.
`number_color`	The same as in `pheatmap`.
`fontsize_number`	The same as in `pheatmap`.
`gaps_row`	The same as in `pheatmap`.
`gaps_col`	The same as in `pheatmap`.
`labels_row`	The same as in `pheatmap`.
`labels_col`	The same as in `pheatmap`.
`filename`	Not supported.
`width`	Not supported.
`height`	Not supported.
`silent`	Not supported.
`na_col`	The same as in `pheatmap`.
`name`	Name of the heatmap. This argument is passed to `Heatmap`.
`heatmap_legend_param`	Pass to `Heatmap`.
`...`	Other arguments passed to `Heatmap`.
`run_draw`	Whether to run `draw()` function to the heatmap object.

Details

This function aims to execute `pheatmap::pheatmap` code purely with ComplexHeatmap.

Value

A `Heatmap-class` object.

See Also

See `https://jokergoo.github.io/2020/05/06/translate-from-pheatmap-to-complexheatmap/`

`compare_pheatmap` that compares heatmaps between `pheatmap::pheatmap()` and `ComplexHeatmap::pheatmap()`.

Examples

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

pindex *Get Values in a Matrix by Pair-wise Indices*

Description

Get Values in a Matrix by Pair-wise Indices

Usage

```
pindex(m, i, j)
```

Arguments

m	A matrix or a 3-dimension array.
i	Row indices or the indices in the first dimension.
j	Column indicies or the indices in the second dimension.

Value

If `m` is a matrix, the value returned is a vector `c(m[i1,j1],m[i2,j2],...)`‘.

If `m` is an array, the value returned is a matrix `rbind(m[i1,j1,],m[i2,j2,],...)`‘.

Examples

```
m = matrix(rnorm(100), 10)
m2 = m[m > 0]
ind = do.call("rbind", lapply(1:10, function(ci) {
    i = which(m[, ci] > 0)
    cbind(i = i, j = rep(ci, length(i)))
}))
pindex(m, ind[, 1], ind[, 2])
identical(pindex(m, ind[, 1], ind[, 2]), m[m > 0])

# 3d array
arr = array(1:27, dim = c(3, 3, 3))
pindex(arr, 1:2, 2:3)
identical(pindex(arr, 1:2, 2:3),
   rbind(arr[1, 2, ], arr[2, 3, ]))
```

plot.Heatmap — *Draw heatmap*

Description

Draw heatmap

Usage

```
## S3 method for class 'Heatmap'
plot(x, ...)
```

Arguments

x	A Heatmap-class object.
...	All pass to draw,Heatmap-method.

Examples

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

plot.HeatmapAnnotation — *Draw heatmap annotations*

Description

Draw heatmap annotations

Usage

```
## S3 method for class 'HeatmapAnnotation'
plot(x, ...)
```

Arguments

x	A HeatmapAnnotation-class object.
...	All pass to draw,HeatmapList-method.

Examples

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

plot.HeatmapList *Draw heatmap*

Description

Draw heatmap

Usage

```
## S3 method for class 'HeatmapList'
plot(x, ...)
```

Arguments

x	A HeatmapList-class object.
...	All pass to draw,HeatmapList-method.

Examples

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

prepare-Heatmap-method *Prepare the Heatmap*

Description

Prepare the Heatmap

Usage

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

Arguments

object	A Heatmap-class object.
process_rows	Whether to process rows of the heatmap.
process_columns	Whether to process columns of the heatmap.

Details

The preparation of the heatmap includes following steps:

- making clustering on rows (by calling `make_row_cluster,Heatmap-method`)
- making clustering on columns (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

The `Heatmap-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

`print.comb_mat` *Print the comb_mat Object*

Description

Print the comb_mat Object

Usage

```
## S3 method for class 'comb_mat'
print(x, ...)
```

Arguments

`x`	A combination matrix returned by `make_comb_mat`.
`...`	Other arguments

Examples

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

restore_matrix *Restore the index vector to index matrix in layer_fun*

Description

Restore the index vector to index matrix in layer_fun

Usage

```
restore_matrix(j, i, x, y)
```

Arguments

j	Column indices directly from `layer_fun`.
i	Row indices directly from `layer_fun`.
x	Position on x-direction directly from `layer_fun`.
y	Position on y-direction directly from `layer_fun`.

Details

The values that are sent to `layer_fun` are all vectors (for the vectorization of the grid graphic functions), however, the heatmap slice where `layer_fun` is applied to, is still represented by a matrix, thus, it would be very convinient if all the arguments in `layer_fun` can be converted to the sub-matrix for the current slice. Here, as shown in above example, `restore_matrix` does the job. `restore_matrix` directly accepts the first four argument in `layer_fun` and returns an index matrix, where rows and columns correspond to the rows and columns in the current slice, from top to bottom and from left to right. The values in the matrix are the natural order of e.g. vector `j` in current slice.

For following code:

```
Heatmap(small_mat, name = "mat", col = col_fun,
    row_km = 2, column_km = 2,
    layer_fun = function(j, i, x, y, w, h, fill) {
        ind_mat = restore_matrix(j, i, x, y)
        print(ind_mat)
    }
)
```

The first output which is for the top-left slice:

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15
```

As you see, this is a three-row and five-column index matrix where the first row corresponds to the top row in the slice. The values in the matrix correspond to the natural index (i.e. 1, 2, ...) in `j`, `i`, `x`, `y`, ... in `layer_fun`. Now, if we want to add values on the second column in the top-left slice, the code which is put inside `layer_fun` would look like:

```
for(ind in ind_mat[, 2]) {
    grid.text(small_mat[i[ind], j[ind]], x[ind], y[ind], ...)
}
```

Examples

```
set.seed(123)
mat = matrix(rnorm(81), nr = 9)
Heatmap(mat, row_km = 2, column_km = 2,
    layer_fun = function(j, i, x, y, width, height, fill) {
       ind_mat = restore_matrix(j, i, x, y)
       print(ind_mat)
})

set.seed(123)
mat = matrix(round(rnorm(81), 2), nr = 9)
Heatmap(mat, row_km = 2, column_km = 2,
    layer_fun = function(j, i, x, y, width, height, fill) {
       ind_mat = restore_matrix(j, i, x, y)
       ind = unique(c(ind_mat[2, ], ind_mat[, 3]))
       grid.text(pindex(mat, i[ind], j[ind]), x[ind], y[ind])
})
```

`re_size-HeatmapAnnotation-method`
Resize the Width or Height of Heatmap Annotations

Description

Resize the Width or Height of Heatmap Annotations

Usage

```
## S4 method for signature 'HeatmapAnnotation'
re_size(object,
    annotation_height = NULL,
    annotation_width = NULL,
    height = NULL,
    width = NULL,
    simple_anno_size = object@param$simple_anno_size,
    simple_anno_size_adjust = object@param$simple_anno_size_adjust)
```

Arguments

`object`	A `HeatmapAnnotation-class` object.
`annotation_height`	A vector of of annotation heights in `unit` class.
`annotation_width`	A vector of of annotation widths in `unit` class.
`height`	The height of the complete heatmap annotation.
`width`	The width of the complete heatmap annotation.
`simple_anno_size`	The size of one line of the simple annotation.
`simple_anno_size_adjust`	Whether adjust the size of the simple annotation?

Details

The function only adjust height for column annotations and width for row annotations.

The basic rules are (take `height` and `annotation_height` for example:

1. If `annotation_height` is set and all `annotation_height` are absolute units, `height` is ignored. 2. If `annotation_height` contains non-absolute units, `height` also need to be set and the non-absolute units should be set in a simple form such as 1:10 or `unit(1,"null")`. 3. `simple_anno_size` is only used when `annotation_height` is NULL. 4. If only `height` is set, non-simple annotation is adjusted while keeps simple anntation unchanged. 5. If only `height` is set and all annotations are simple annotations, all anntations are adjusted, and `simple_anno_size` is disabled. 6. If `simple_anno_size_adjust` is `FALSE`, the size of the simple annotations will not change.

Examples

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

`rowAnnotation` *Construct Row Annotations*

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

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

`row_anno_barplot` *Barplots as Row Annotation*

Description

Barplots as Row Annotation

Usage

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

Arguments

`...`	pass to `anno_barplot`.

Details

A wrapper of `anno_barplot` with pre-defined `which` to `row`.

You can directly use `anno_barplot` for row annotation if you call it in `rowAnnotation`.

Value

See help page of `anno_barplot`.

Examples

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

`row_anno_boxplot` *Boxplots as Row Annotation*

Description

Boxplots as Row Annotation

Usage

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

Arguments

`...`	pass to `anno_boxplot`.

Details

A wrapper of `anno_boxplot` with pre-defined `which` to `row`.

You can directly use `anno_boxplot` for row annotation if you call it in `rowAnnotation`.

Value

See help page of `anno_boxplot`.

Examples

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

`row_anno_density` *Density as Row Annotation*

Description

Density as Row Annotation

Usage

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

Arguments

`...`	pass to `anno_density`.

Details

A wrapper of `anno_density` with pre-defined `which` to `row`.

You can directly use `anno_density` for row annotation if you call it in `rowAnnotation`.

Value

See help page of `anno_density`.

Examples

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

`row_anno_histogram` *Histograms as Row Annotation*

Description

Histograms as Row Annotation

Usage

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

Arguments

`...`	pass to `anno_histogram`.

Details

A wrapper of `anno_histogram` with pre-defined `which` to `row`.

You can directly use `anno_histogram` for row annotation if you call it in `rowAnnotation`.

Value

See help page of `anno_histogram`.

Examples

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

`row_anno_points` *Points as Row Annotation*

Description

Points as Row Annotation

Usage

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

Arguments

`...`	pass to `anno_points`.

Details

A wrapper of `anno_points` with pre-defined `which` to `row`.

You can directly use `anno_points` for row annotation if you call it in `rowAnnotation`.

Value

See help page of `anno_points`.

Examples

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

`row_anno_text` *Text as Row Annotation*

Description

Text as Row Annotation

Usage

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

Arguments

`...`	pass to `anno_text`.

Details

A wrapper of `anno_text` with pre-defined `which` to `row`.

You can directly use `anno_text` for row annotation if you call it in `rowAnnotation`.

Value

See help page of `anno_text`.

Examples

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

`row_dend-dispatch` *Method dispatch page for row_dend*

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

The format of the returned object depends on whether rows/columns of the heatmaps are split.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
mat = matrix(rnorm(100), 10)
ht = Heatmap(mat)
ht = draw(ht)
row_dend(ht)
ht = Heatmap(mat, row_km = 2)
ht = draw(ht)
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, name = NULL)
```

Arguments

`object`	A `HeatmapList-class` object.
`name`	Name of a specific heatmap.

Value

The format of the returned object depends on whether rows/columns of the heatmaps are split.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
mat = matrix(rnorm(100), 10)
ht_list = Heatmap(mat) + Heatmap(mat)
ht_list = draw(ht_list)
row_dend(ht_list)
ht_list = Heatmap(mat, row_km = 2) + Heatmap(mat)
ht_list = draw(ht_list)
row_dend(ht_list)
ht_list = Heatmap(mat, row_km = 2) %v% Heatmap(mat)
ht_list = draw(ht_list)
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

The format of the returned object depends on whether rows/columns of the heatmaps are split.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
mat = matrix(rnorm(100), 10)
ht = Heatmap(mat)
ht = draw(ht)
row_order(ht)
ht = Heatmap(mat, row_km = 2)
ht = draw(ht)
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, name = NULL)
```

Arguments

`object`	A `HeatmapList-class` object.
`name`	Name of a specific heatmap.

Value

The format of the returned object depends on whether rows/columns of the heatmaps are split.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

```
mat = matrix(rnorm(100), 10)
ht_list = Heatmap(mat) + Heatmap(mat)
ht_list = draw(ht_list)
row_order(ht_list)
ht_list = Heatmap(mat, row_km = 2) + Heatmap(mat)
ht_list = draw(ht_list)
row_order(ht_list)
ht_list = Heatmap(mat, row_km = 2) %v% Heatmap(mat)
ht_list = draw(ht_list)
row_order(ht_list)
```

`set_component_height-Heatmap-method`
Set Height of Heatmap Component

Description

Set Height of Heatmap Component

Usage

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

Arguments

`object`	A `Heatmap-class` object.
`k`	Which column component? The value should a numeric index or the name of the corresponding column component. See **Detials**.
`v`	Height of the component, a `unit` object.

Details

All column components are: `column_title_top`, `column_dend_top`, `column_names_top`, `column_anno_top`, `heatmap_body`, `column_anno_bottom`, `column_names_bottom`, `column_dend_bottom`, `column_title_bottom`.

This function is only for internal use.

Value

The `Heatmap-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

---

set_component_width-Heatmap-method

Set Width of Heatmap Component

Description

Set Width of Heatmap Component

Usage

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

Arguments

`object`	A `Heatmap-class` object.
`k`	Which row component? The value should a numeric index or the name of the corresponding row component. See **Detials**.
`v`	width of the component, a `unit` object.

Details

All row components are: `row_title_left`, `row_dend_left`, `row_names_left`, `row_anno_left`, `heatmap_body`, `row_anno_right`, `row_names_right`, `row_dend_right`, `row_title_right`.

This function is only for internal use.

Value

The Heatmap-class object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

---

set_name *Set Names*

Description

Set Names

Usage

```
set_name(m)
```

Arguments

m	A combination matrix returned by make_comb_mat.

Value

A vector of set names.

Examples

```
set.seed(123)
lt = list(a = sample(letters, 10),
          b = sample(letters, 15),
          c = sample(letters, 20))
m = make_comb_mat(lt)
set_name(m)
```

set_nameAssign *Modify Set Names*

Description

Modify Set Names

Usage

```
set_name(x, ...) <- value
```

Arguments

`x`	A combination matrix returned by `make_comb_mat`.
`value`	New set names.
`...`	Other arguments.

Examples

```
set.seed(123)
lt = list(a = sample(letters, 10),
          b = sample(letters, 15),
          c = sample(letters, 20))
m = make_comb_mat(lt)
set_name(m) = c("A", "B", "C")
m
```

set_size *Set Sizes*

Description

Set Sizes

Usage

```
set_size(m)
```

Arguments

`m`	A combination matrix returned by `make_comb_mat`.

Value

A vector of set sizes.

Examples

```
set.seed(123)
lt = list(a = sample(letters, 10),
          b = sample(letters, 15),
          c = sample(letters, 20))
m = make_comb_mat(lt)
set_size(m)
```

---

show-AnnotationFunction-method

Print the AnnotationFunction Object

Description

Print the AnnotationFunction Object

Usage

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

Arguments

`object`	The `AnnotationFunction-class` object.

Examples

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

---

show-ColorMapping-method

Print the ColorMapping Object

Description

Print the 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` *Method dispatch page for show*

Description

Method dispatch page for `show`.

Dispatch

`show` can be dispatched on following classes:

- `show,ColorMapping-method`, `ColorMapping-class` class method
- `show,SingleAnnotation-method`, `SingleAnnotation-class` class method
- `show,AnnotationFunction-method`, `AnnotationFunction-class` class method
- `show,HeatmapList-method`, `HeatmapList-class` class method
- `show,HeatmapAnnotation-method`, `HeatmapAnnotation-class` class method
- `show,Heatmap-method`, `Heatmap-class` class method

Examples

```
# no example
NULL
```

`show-Heatmap-method` *Draw the Single Heatmap with Defaults*

Description

Draw the Single Heatmap with Defaults

Usage

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

Arguments

`object`	A `Heatmap-class` object.

Details

It actually 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

The `HeatmapList-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

Examples

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

`show-HeatmapAnnotation-method` *Print the HeatmapAnnotation object*

Description

Print the HeatmapAnnotation 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.

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` *Constructor Method for SingleAnnotation Class*

Description

Constructor Method for SingleAnnotation Class

Usage

```
SingleAnnotation(name, value, col, fun,
    label = NULL,
    na_col = "grey",
    which = c("column", "row"),
    show_legend = TRUE,
    gp = gpar(col = NA),
    border = FALSE,
    legend_param = list(),
    show_name = TRUE,
    name_gp = gpar(fontsize = 12),
    name_offset = NULL,
    name_side = ifelse(which == "column", "right", "bottom"),
    name_rot = NULL,
    simple_anno_size = ht_opt$simple_anno_size,
    width = NULL, height = NULL)
```

Arguments

`name`	Name for the annotation. If it is not specified, an internal name is assigned.
`value`	A vector or a matrix of discrete or continuous values.
`col`	Colors corresponding to `value`. If the mapping is discrete, the value of `col` should be a named vector; If the mapping is continuous, the value of `col` should be a color mapping function.
`fun`	A user-defined function to add annotation graphics. The argument of this function should be at least a vector of index that corresponds to rows or columns. Normally the function should be constructed by `AnnotationFunction` if you want the annotation supports splitting. See **Details** for more explanation.
`label`	Label for the annotation. By default is the annotation name.
`na_col`	Color for `NA` values in the simple annotations.
`which`	Whether the annotation is a row annotation or a column annotation?
`show_legend`	If it is a simple annotation, whether show legend in the final heatmap?
`gp`	Since simple annotation is represented as rows of grids. This argument controls graphic parameters for the simple annotation. The `fill` parameter is ignored here.
`border`	border, only work for simple annotation
`legend_param`	Parameters for the legend. See `color_mapping_legend,ColorMapping-method` for all possible options.
`show_name`	Whether show annotation name?
`name_gp`	Graphic parameters for annotation name.
`name_offset`	Offset to the annotation, a `unit` object.
`name_side`	'right' and 'left' for column annotations and 'top' and 'bottom' for row annotations
`name_rot`	Rotation of the annotation name.

`simple_anno_size`	size of the simple annotation.
`width`	The width of the plotting region (the viewport) that the annotation is drawn. If it is a row annotation, the width must be an absolute unit.
`height`	The height of the plotting region (the viewport) that the annotation is drawn. If it is a column annotation, the width must be an absolute unit.

Details

A single annotation is a basic unit of complex heatmap annotations where the heamtap annotations are always a list of single annotations. An annotation can be simply heatmap-like (here we call it simple annotation) or more complex like points, lines, boxes (for which we call it complex annotation).

In the `SingleAnnotation` constructor, `value`, `col`, `na_col` are used to construct a `anno_simple` annotation funciton which is generated internally by `AnnotationFunction`. The legend of the simple annotation can be automatcally generated,

For construcing a complex annotation, users need to use `fun` which is a user-defind function. Normally it is constucted by `AnnotationFunction`. One big advantage for using `AnnotationFunction` is the annotation function or the graphics drawn by the annotation function can be split according to row splitting or column splitting of the heatmap. Users can also provide a "pure" function which is a normal R function for the `fun` argument. The function only needs one argument which is a vector of index for rows or columns depending whether it is a row annotation or column annotation. The other two optional arguments are the current slice index and total number of slices. See **Examples** section for an example. If it is a normal R function, it will be constructed into the `AnnotationFunction-class` object internally.

The `SingleAnnotation-class` is a simple wrapper on top of `AnnotationFunction-class` only with annotation name added.

The class also stored the "extended area" relative to the area for the annotation graphics. The extended areas are those created by annotation names and axes.

Value

A `SingleAnnotation-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

There are following built-in annotation functions that can be directly used to generate complex annotations: `anno_simple`, `anno_points`, `anno_lines`, `anno_barplot`, `anno_histogram`, `anno_boxplot`, `anno_density`, `anno_text`, `anno_joyplot`, `anno_horizon`, `anno_image`, `anno_block`, `anno_summary` and `anno_mark`.

Examples

```
ha = SingleAnnotation(value = 1:10)
draw(ha, test = "single column annotation")

m = cbind(1:10, 10:1)
colnames(m) = c("a", "b")
ha = SingleAnnotation(value = m)
draw(ha, test = "matrix as column annotation")

anno = anno_barplot(matrix(nc = 2, c(1:10, 10:1)))
ha = SingleAnnotation(fun = anno)
draw(ha, test = "anno_barplot as input")

fun = local({
    # because there variables outside the function for use, we put it a local environment
    value = 1:10
    function(index, k = 1, n = 1) {
       pushViewport(viewport(xscale = c(0.5, length(index) + 0.5), yscale = range(value)))
         grid.points(seq_along(index), value[index])
         grid.rect()
         if(k == 1) grid.yaxis()
         popViewport()
    }
})
ha = SingleAnnotation(fun = fun, height = unit(4, "cm"))
draw(ha, index = 1:10, test = "self-defined function")
```

SingleAnnotation-class

Class for a Single Annotation

Description

Class for a Single Annotation

Details

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

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

---

size.AnnotationFunction

Size of the AnnotationFunction Object

Description

Size of the AnnotationFunction Object

Usage

```
## S3 method for class 'AnnotationFunction'
size(x, ...)
```

Arguments

`x`	The AnnotationFunction-class object.
`...`	Other arguments.

Details

It returns the width if it is a row annotation and the height if it is a column annotation.

Internally used.

Examples

```
anno = anno_points(1:10)
ComplexHeatmap:::size(anno)
anno = anno_points(1:10, which = "row")
ComplexHeatmap:::size(anno)
```

size.HeatmapAnnotation — *Size of the HeatmapAnnotation Object*

Description

Size of the HeatmapAnnotation Object

Usage

```
## S3 method for class 'HeatmapAnnotation'
size(x, ...)
```

Arguments

`x`	The `HeatmapAnnotation-class` object.
`...`	Other arguments.

Details

It returns the width if it is a row annotation and the height if it is a column annotation.

Internally used.

Examples

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

size.SingleAnnotation — *Size of the SingleAnnotation Object*

Description

Size of the SingleAnnotation Object

Usage

```
## S3 method for class 'SingleAnnotation'
size(x, ...)
```

Arguments

`x`	The `SingleAnnotation-class` object.
`...`	Other arguments.

Details

It returns the width if it is a row annotation and the height if it is a column annotation.

Internally used.

Examples

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

`sizeAssign.AnnotationFunction`

Assign the Size to the AnnotationFunction Object

Description

Assign the Size to the AnnotationFunction Object

Usage

```
## S3 replacement method for class 'AnnotationFunction'
size(x, ...) <- value
```

Arguments

`x`	The `AnnotationFunction-class` object.
`value`	A `unit` object.
`...`	Other arguments.

Details

It assigns to the width if it is a row annotation and the height if it is a column annotation.

Internally used.

Examples

```
anno = anno_points(1:10)
ComplexHeatmap:::size(anno) = unit(4, "cm")
ComplexHeatmap:::size(anno)
```

sizeAssign.HeatmapAnnotation

Assign the Size to the HeatmapAnnotation Object

Description

Assign the Size to the HeatmapAnnotation Object

Usage

```
## S3 replacement method for class 'HeatmapAnnotation'
size(x, ...) <- value
```

Arguments

x	The HeatmapAnnotation-class object.
value	A unit object.
...	Other arguments.

Details

It assigns the width if it is a row annotation and the height if it is a column annotation.

Internally used.

Examples

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

sizeAssign.SingleAnnotation

Assign the Size to the SingleAnnotation Object

Description

Assign the Size to the SingleAnnotation Object

Usage

```
## S3 replacement method for class 'SingleAnnotation'
size(x, ...) <- value
```

Arguments

`x`	The `SingleAnnotation-class` object.
`value`	A `unit` object.
`...`	Other arguments.

Details

It assigns to the width if it is a row annotation and the height if it is a column annotation.

Internally used.

Examples

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

`smartAlign2` *Adjust positions of rectanglar shapes*

Description

Adjust positions of rectanglar shapes

Usage

```
smartAlign2(start, end, range, plot = FALSE)
```

Arguments

`start`	position which corresponds to the start (bottom or left) of the rectangle-shapes.
`end`	position which corresponds to the end (top or right) of the rectanglar shapes.
`range`	data ranges (the minimal and maximal values)
`plot`	Whether plot the correspondance between the original positions and the adjusted positions. Only for testing.

Details

This is an improved version of the `smartAlign`.

It adjusts the positions of the rectangular shapes to make them do not overlap

Examples

```
range = c(0, 10)
pos1 = rbind(c(1, 2), c(5, 7))
smartAlign2(pos1, range = range, plot = TRUE)

range = c(0, 10)
pos1 = rbind(c(-0.5, 2), c(5, 7))
smartAlign2(pos1, range = range, plot = TRUE)

pos1 = rbind(c(-1, 2), c(3, 4), c(5, 6), c(7, 11))
pos1 = pos1 + runif(length(pos1), max = 0.3, min = -0.3)
omfrow = par("mfrow")
par(mfrow = c(3, 3))
for(i in 1:9) {
    ind = sample(4, 4)
    smartAlign2(pos1[ind, ], range = range, plot = TRUE)
}
par(mfrow = omfrow)

pos1 = rbind(c(3, 6), c(4, 7))
smartAlign2(pos1, range = range, plot = TRUE)

pos1 = rbind(c(1, 8), c(3, 10))
smartAlign2(pos1, range = range, plot = TRUE)
```

`str.comb_mat` *str method*

Description

str method

Usage

```
## S3 method for class 'comb_mat'
str(object, ...)
```

Arguments

`object`	A combination matrix returned by `make_comb_mat`.
`...`	Other arguments.

Examples

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

`subset_gp` *Subset a gpar Object*

Description

Subset a gpar Object

Usage

```
subset_gp(gp, i)
```

Arguments

gp	A gpar object.
i	A vector of indices.

Value

A gpar object.

Examples

```
gp = gpar(col = 1:10, fill = 1)
subset_gp(gp, 1:5)
```

`subset_matrix_by_row` *Subset the Matrix by Rows*

Description

Subset the Matrix by Rows

Usage

```
subset_matrix_by_row(x, i)
```

Arguments

x	A matrix.
i	The row indices.

Details

Mainly used for constructing the AnnotationFunction-class object.

Examples

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

subset_no *Do not do subseting*

Description

Do not do subseting

Usage

```
subset_no(x, i)
```

Arguments

x	A vector.
i	The indices.

Details

Mainly used for constructing the `AnnotationFunction-class` object.

Examples

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

subset_vector *Subset the vector*

Description

Subset the vector

Usage

```
subset_vector(x, i)
```

Arguments

x	A vector.
i	The indices.

Details

Mainly used for constructing the `AnnotationFunction-class` object.

Examples

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

`summary.Heatmap` *Print the Summary of a Heatmap*

Description

Print the Summary of a Heatmap

Usage

```
## S3 method for class 'Heatmap'
summary(object, ...)
```

Arguments

`object`	A `Heatmap-class` object.
`...`	Other arguments.

Examples

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

`summary.HeatmapList` *Summary of a Heatmap List*

Description

Summary of a Heatmap List

Usage

```
## S3 method for class 'HeatmapList'
summary(object, ...)
```

Arguments

`object`	A `HeatmapList-class` object.
`...`	Other arguments.

Examples

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

`t.comb_mat` *Transpost the Combination Matrix*

Description

Transpost the Combination Matrix

Usage

```
## S3 method for class 'comb_mat'
t(x)
```

Arguments

`x`	A combination matrix returned by `make_comb_mat`.

Examples

```
set.seed(123)
lt = list(a = sample(letters, 10),
          b = sample(letters, 15),
          c = sample(letters, 20))
m = make_comb_mat(lt)
t(m)
```

`test_alter_fun` *Test alter_fun for oncoPrint()*

Description

Test alter_fun for oncoPrint()

Usage

```
test_alter_fun(fun, type, asp_ratio = 1)
```

Arguments

`fun`	The `alter_fun` for `oncoPrint`. The value can be a list of functions or a single function. See `https://jokergoo.github.io/ComplexHeatmap-reference/book/oncoprint.html#define-the-alter-fun`
`type`	A vector of alteration types. It is only used when `fun` is a single function.
`asp_ratio`	The aspect ratio (width/height) for the small rectangles.

Details

This function helps you to have a quick view of how the graphics for each alteration type and combinations look like.

Examples

```
alter_fun = list(
mut1 = function(x, y, w, h) grid.rect(x, y, w, h, gp = gpar(fill = "red", col = NA)),
mut2 = function(x, y, w, h) grid.rect(x, y, w, h, gp = gpar(fill = "blue", col = NA)),
mut3 = function(x, y, w, h) grid.rect(x, y, w, h, gp = gpar(fill = "yellow", col = NA)),
mut4 = function(x, y, w, h) grid.rect(x, y, w, h, gp = gpar(fill = "purple", col = NA)),
mut5 = function(x, y, w, h) grid.rect(x, y, w, h, gp = gpar(lwd = 2)),
mut6 = function(x, y, w, h) grid.points(x, y, pch = 16),
mut7 = function(x, y, w, h) grid.segments(x - w*0.5, y - h*0.5, x + w*0.5, y + h*0.5, gp = gpar(lwd = 2))
)
test_alter_fun(alter_fun)
```

`unify_mat_list` *Unify a List of Matrix*

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
```

UpSet *Make the UpSet plot*

Description

Make the UpSet plot

Usage

```
UpSet(m,
    comb_col = "black",
    pt_size = unit(3, "mm"), lwd = 2,
    bg_col = "#F0F0F0", bg_pt_col = "#CCCCCC",
    set_order = order(set_size(m), decreasing = TRUE),
    comb_order = if(attr(m, "param")$set_on_rows) {
    order.comb_mat(m[set_order, ], decreasing = TRUE)
    } else {
    order.comb_mat(m[, set_order], decreasing = TRUE)
    },
    top_annotation = upset_top_annotation(m),
    right_annotation = upset_right_annotation(m),
    left_annotation = NULL,
    row_names_side = "left",
    ...)
```

Arguments

m	A combination matrix returned by make_comb_mat. The matrix can be transposed to switch the position of sets and combination sets.
comb_col	The color for the dots representing combination sets.
pt_size	The point size for the dots representing combination sets.
lwd	The line width for the combination sets.
bg_col	Color for the background rectangles.
bg_pt_col	Color for the dots representing the set is not selected.
set_order	The order of sets.
comb_order	The order of combination sets.

`top_annotation`	A `HeatmapAnnotation` object on top of the combination matrix.
`left_annotation`	A `HeatmapAnnotation` object on top of the combination matrix.
`right_annotation`	A `HeatmapAnnotation` object on the right of the combination matrix.
`row_names_side`	The side of row names.
`...`	Other arguments passed to `Heatmap`.

Details

By default, the sets are on rows and combination sets are on columns. The positions of the two types of sets can be switched by transposing the matrix.

When sets are on rows, the default top annotation is the barplot showing the size of each combination sets and the default right annotation is the barplot showing the size of the sets. The annotations are simply constructed by `HeatmapAnnotation` and `anno_barplot` with some parameters pre-set. Users can check the source code of `upset_top_annotation` and `upset_right_annotation` to find out how the annotations are defined.

To change or to add annotations, users just need to define a new `HeatmapAnnotation` object. E.g. if we want to change the side of the axis and name on top annotation:

```
Upset(..., top_annotation =
    HeatmapAnnotation(
       "Intersection size" = anno_barplot(
           comb_size(m),
           border = FALSE,
           gp = gpar(fill = "black"),
           height = unit(2, "cm"),
           axis_param = list(side = "right")
       ),
       annotation_name_side = "right",
       annotation_name_rot = 0)
)
```

To add more annotations on top, users just add it in `HeatmapAnnotation`:

```
Upset(..., top_annotation =
    HeatmapAnnotation(
       "Intersection size" = anno_barplot(
           comb_size(m),
           border = FALSE,
           gp = gpar(fill = "black"),
           height = unit(2, "cm"),
           axis_param = list(side = "right")
       ),
       "anno1" = anno_points(...),
       "anno2" = some_vector,
       annotation_name_side = "right",
            annotation_name_rot = 0)
    )
```

And so is for the right annotations.

UpSet returns a Heatmap-class object, which means, you can add it with other heatmaps and annotations by + or %v%.

Examples

```
set.seed(123)
lt = list(a = sample(letters, 10),
          b = sample(letters, 15),
          c = sample(letters, 20))
m = make_comb_mat(lt)
UpSet(m)
UpSet(t(m))

m = make_comb_mat(lt, mode = "union")
UpSet(m)
UpSet(m, comb_col = c(rep(2, 3), rep(3, 3), 1))

# compare two UpSet plots
set.seed(123)
lt1 = list(a = sample(letters, 10),
          b = sample(letters, 15),
          c = sample(letters, 20))
m1 = make_comb_mat(lt1)
set.seed(456)
lt2 = list(a = sample(letters, 10),
          b = sample(letters, 15),
          c = sample(letters, 20))
m2 = make_comb_mat(lt2)

max1 = max(c(set_size(m1), set_size(m2)))
max2 = max(c(comb_size(m1), comb_size(m2)))

UpSet(m1, top_annotation = upset_top_annotation(m1, ylim = c(0, max2)),
    right_annotation = upset_right_annotation(m1, ylim = c(0, max1)),
    column_title = "UpSet1") +
UpSet(m2, top_annotation = upset_top_annotation(m2, ylim = c(0, max2)),
    right_annotation = upset_right_annotation(m2, ylim = c(0, max1)),
    column_title = "UpSet2")
```

---

upset_left_annotation *UpSet Left Annotation*

Description

UpSet Left Annotation

Usage

```
upset_left_annotation(m,
    gp = gpar(fill = "black"),
    axis_param = list(direction = "reverse"),
    width = unit(ifelse(set_on_rows, 2, 3), "cm"),
    show_annotation_name = TRUE,
    annotation_name_gp = gpar(),
    annotation_name_offset = NULL,
    annotation_name_side = "bottom",
    annotation_name_rot = NULL,
    ...)
```

Arguments

`m`	A combination matrix which is as same as the one for `UpSet`.
`gp`	Graphic parameters for bars.
`axis_param`	Parameters for axis.
`width`	Width of the left annotation.
`show_annotation_name`	Whether show annotation names?
`annotation_name_gp`	Graphic parameters for anntation names.
`annotation_name_offset`	Offset to the annotation name, a `unit` object.
`annotation_name_side`	Side of the annotation name.
`annotation_name_rot`	Rotation of the annotation name, it can only take values in `c(00,90,180,270)`.
`...`	Passed to `anno_barplot`, e.g. to set `add_numbers`.

Examples

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

`upset_right_annotation` *Default UpSet Right Annotation*

Description

Default UpSet Right Annotation

Usage

```
upset_right_annotation(m,
    gp = gpar(fill = "black"),
    width = unit(ifelse(set_on_rows, 2, 3), "cm"),
    show_annotation_name = TRUE,
    annotation_name_gp = gpar(),
    annotation_name_offset = NULL,
    annotation_name_side = "bottom",
    annotation_name_rot = NULL,
    ...)
```

Arguments

`m`	A combination matrix which is as same as the one for `UpSet`.
`gp`	Graphic parameters for bars.
`width`	Width of the right annotation.
`show_annotation_name`	Whether show annotation names?
`annotation_name_gp`	Graphic parameters for anntation names.
`annotation_name_offset`	Offset to the annotation name, a `unit` object.
`annotation_name_side`	Side of the annotation name.
`annotation_name_rot`	Rotation of the annotation name, it can only take values in `c(00,90,180,270)`.
`...`	Passed to `anno_barplot`, e.g. to set `add_numbers`.

Details

The default right annotation is actually barplot implemented by `anno_barplot`. For how to set the right annotation or left annotation in `UpSet`, please refer to `UpSet`.

If you want to use `decorate_annotation` function, the annotation name for the "sets" is `set_size` and the annotation name for the "intersection sets" are `intersection_size` and if under the union mode, it is `union_size`.

Examples

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

`upset_top_annotation` *Default UpSet Top Annotation*

Description

Default UpSet Top Annotation

Usage

```
upset_top_annotation(m,
    gp = gpar(fill = "black"),
    height = unit(ifelse(set_on_rows, 3, 2), "cm"),
    show_annotation_name = TRUE,
    annotation_name_gp = gpar(),
    annotation_name_offset = NULL,
    annotation_name_side = "left",
    annotation_name_rot = 0,
    ...)
```

Arguments

`m`	A combination matrix which is as same as the one for `UpSet`.
`gp`	Graphic parameters for bars.
`height`	The height of the top annotation.
`show_annotation_name`	Whether show annotation names?
`annotation_name_gp`	Graphic parameters for anntation names.
`annotation_name_offset`	Offset to the annotation name, a `unit` object.
`annotation_name_side`	Side of the annotation name.
`annotation_name_rot`	Rotation of the annotation name, it can only take values in `c(00,90,180,270)`.
`...`	Passed to `anno_barplot`.

Details

The default top annotation is actually barplot implemented by `anno_barplot`. For how to set the top annotation or bottom annotation in `UpSet`, please refer to `UpSet`.

If you want to use `decorate_annotation` function, the annotation name for the "sets" is `set_size` and the annotation name for the "intersection sets" are `intersection_size` and if under the union mode, it is `union_size`.

Examples

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

`width.AnnotationFunction` *Width of the AnnotationFunction Object*

Description

Width of the AnnotationFunction Object

Usage

```
## S3 method for class 'AnnotationFunction'
width(x, ...)
```

Arguments

x	A `AnnotationFunction-class` object.
...	Other arguments.

Details

Internally used.

Examples

```
anno = anno_points(1:10)
ComplexHeatmap:::width(anno)
anno = anno_points(1:10, which = "row")
ComplexHeatmap:::width(anno)
```

`width.Heatmap` *Width of the Heatmap*

Description

Width of the Heatmap

Usage

```
## S3 method for class 'Heatmap'
width(x, ...)
```

Arguments

`x`	The `HeatmapList-class` object returned by `draw,Heatmap-method`.
`...`	Other arguments.

Examples

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

`width.HeatmapAnnotation`

Width of the HeatmapAnnotation Object

Description

Width of the HeatmapAnnotation Object

Usage

```
## S3 method for class 'HeatmapAnnotation'
width(x, ...)
```

Arguments

`x`	The `HeatmapAnnotation-class` object.
`...`	Other arguments.

Details

Internally used.

Examples

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

width.HeatmapList *Width of the Heatmap List*

Description

Width of the Heatmap List

Usage

```
## S3 method for class 'HeatmapList'
width(x, ...)
```

Arguments

x	The HeatmapList-class object returned by draw,HeatmapList-method.
...	Other arguments.

Examples

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

width.Legends *Width of the Legends*

Description

Width of the Legends

Usage

```
## S3 method for class 'Legends'
width(x, ...)
```

Arguments

x	The grob object returned by Legend or packLegend.
...	Other arguments.

Value

The returned unit x is always in mm.

Examples

```
lgd = Legend(labels = 1:10, title = "foo", legend_gp = gpar(fill = "red"))
ComplexHeatmap:::width(lgd)
```

`width.SingleAnnotation`
Width of the SingleAnnotation Object

Description

Width of the SingleAnnotation Object

Usage

```
## S3 method for class 'SingleAnnotation'
width(x, ...)
```

Arguments

x	The `SingleAnnotation-class` object.
...	Other arguments.

Details

Internally used.

Examples

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

`widthAssign.AnnotationFunction`
Assign the Width to the AnnotationFunction Object

Description

Assign the Width to the AnnotationFunction Object

Usage

```
## S3 replacement method for class 'AnnotationFunction'
width(x, ...) <- value
```

Arguments

x	The `AnnotationFunction-class` object.
...	Other arguments.
value	A `unit` object.

Details

Internally used.

Examples

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

`widthAssign.HeatmapAnnotation`
Assign the Width to the HeatmapAnnotation Object

Description

Assign the Width to the HeatmapAnnotation Object

Usage

```
## S3 replacement method for class 'HeatmapAnnotation'
width(x, ...) <- value
```

Arguments

`x`	The `HeatmapAnnotation-class` object.
`value`	A `unit` object.
`...`	Other arguments.

Details

Internally used.

Examples

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

widthAssign.SingleAnnotation

Assign the Width to the SingleAnnotation Object

Description

Assign the Width to the SingleAnnotation Object

Usage

```
## S3 replacement method for class 'SingleAnnotation'
width(x, ...) <- value
```

Arguments

`x`	The `SingleAnnotation-class` object.
`value`	A `unit` object.
`...`	Other arguments.

Details

Internally used.

Examples

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

widthDetails.annotation_axis

Width for annotation_axis Grob

Description

Width for annotation_axis Grob

Usage

```
## S3 method for class 'annotation_axis'
widthDetails(x)
```

Arguments

`x`	The `annotation_axis` grob returned by `annotation_axis_grob`.

Details

The physical width of the grob can be get by `convertWidth(grobWidth(axis_grob),"mm")`.

Examples

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

`widthDetails.legend` *Grob width for packed_legends*

Description

Grob width for packed_legends

Usage

```
## S3 method for class 'legend'
widthDetails(x)
```

Arguments

`x`	A legend object.

Examples

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

`widthDetails.legend_body` *Grob width for legend_body*

Description

Grob width for legend_body

Usage

```
## S3 method for class 'legend_body'
widthDetails(x)
```

Arguments

`x`	A legend_body object.

Examples

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

---

widthDetails.packed_legends *Grob width for packed_legends*

Description

Grob width for packed_legends

Usage

```
## S3 method for class 'packed_legends'
widthDetails(x)
```

Arguments

`x`	A packed_legends object.

Examples

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

[.AnnotationFunction *Subset an AnnotationFunction Object*

Description

Subset an AnnotationFunction Object

Usage

```
## S3 method for class 'AnnotationFunction'
x[i]
```

Arguments

`x`	An `AnnotationFunction-class` object.
`i`	A vector of indices.

Details

One good thing for designing the AnnotationFunction-class is it can be subsetted, and this is the base for the splitting of the annotations.

Examples

```
anno = anno_simple(1:10)
anno[1:5]
draw(anno[1:5], test = "subset of column annotation")
```

[.comb_mat *Subset the Combination Matrix*

Description

Subset the Combination Matrix

Usage

```
## S3 method for class 'comb_mat'
x[i, j, drop = FALSE]
```

Arguments

x	A combination matrix returned by make_comb_mat.
i	Indices on rows.
j	Indices on columns.
drop	It is always reset to FALSE internally.

Details

If sets are on rows of the combination matrix, the row indices correspond to sets and column indices correspond to combination sets, and if sets are on columns of the combination matrix, rows correspond to the combination sets.

If the index is one-dimension, e.g. x[i], the index always corresponds to the combination sets.

You should not subset by the sets. It will give you wrong combination set size. The subsetting on sets are only used internally.

This subsetting method is mainly for subsetting combination sets, i.e., users can first use comb_size to get the size of each combination set, and filter them by the size.

Examples

```
set.seed(123)
lt = list(a = sample(letters, 10),
          b = sample(letters, 15),
          c = sample(letters, 20))
m = make_comb_mat(lt)
m2 = m[, comb_size(m) >= 3]
comb_size(m2)
m[comb_size(m) >= 3]
```

`[.gridtext` *Subset method of gridtext class*

Description

Subset method of gridtext class

Usage

```
## S3 method for class 'gridtext'
x[index]
```

Arguments

`x`	A vector of labels generated by `gt_render`.
`index`	Index

Details

Internally used.

Examples

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

[.Heatmap *Subset a Heatmap*

Description

Subset a Heatmap

Usage

```
## S3 method for class 'Heatmap'
x[i, j]
```

Arguments

x	A Heatmap-class object.
i	Row indices.
j	Column indices.

Details

This functionality is quite experimental. It should be applied before the layout is initialized.

Examples

```
m = matrix(rnorm(100), nrow = 10)
rownames(m) = letters[1:10]
colnames(m) = LETTERS[1:10]
ht = Heatmap(m)
ht[1:5, ]
ht[1:5]
ht[, 1:5]
ht[1:5, 1:5]
```

[.HeatmapAnnotation *Subset the HeatmapAnnotation object*

Description

Subset the HeatmapAnnotation object

Usage

```
## S3 method for class 'HeatmapAnnotation'
x[i, j]
```

Arguments

`x`	A `HeatmapAnnotation-class` object.
`i`	Index of observations.
`j`	Index of annotations.

Examples

```
ha = HeatmapAnnotation(foo = 1:10, bar = anno_points(10:1),
sth = cbind(1:10, 10:1))
ha[1:5, ]
ha[, c("foo", "bar")]
ha[, 1:2]
ha[1:5, c("foo", "sth")]
```

`[.HeatmapList` *Subset a HeatmapList object*

Description

Subset a HeatmapList object

Usage

```
## S3 method for class 'HeatmapList'
x[i, j]
```

Arguments

`x`	A `HeatmapList-class` object
`i`	row indices
`j`	column indices

Details

If the heatmap list is horizontal, `i` is the row indices and `j` corresponds to heatmap names and single annotation names. and if the heatlist is vertical, `i` corresponds to heatmap/annotation names and `j` is the column indices.

Examples

```
ht_list = Heatmap(matrix(rnorm(100), 10), name = "rnorm") +
  rowAnnotation(foo = 1:10, bar = anno_points(10:1)) +
  Heatmap(matrix(runif(100), 10), name = "runif")
summary(ht_list[1:5, ])
summary(ht_list[1:5, 1])
summary(ht_list[1:5, "rnorm"])
summary(ht_list[1:5, c("rnorm", "foo")])
```

```
ht_list = Heatmap(matrix(rnorm(100), 10), name = "rnorm") %v%
  columnAnnotation(foo = 1:10, bar = anno_points(10:1)) %v%
  Heatmap(matrix(runif(100), 10), name = "runif")
summary(ht_list[, 1:5])
summary(ht_list[1, 1:5])
summary(ht_list["rnorm", 1:5])
summary(ht_list[c("rnorm", "foo"), 1:5])
```

[.SingleAnnotation *Subset an SingleAnnotation Object*

Description

Subset an SingleAnnotation Object

Usage

```
## S3 method for class 'SingleAnnotation'
x[i]
```

Arguments

x	An SingleAnnotation-class object.
i	A vector of indices.

Details

The SingleAnnotation class object is subsetable only if the containing AnnotationFunction-class object is subsetable. All the anno_* functions are subsetable, so if the SingleAnnotation object is constructed by one of these functions, it is also subsetable.

Examples

```
ha = SingleAnnotation(value = 1:10)
ha[1:5]
draw(ha[1:5], test = "ha[1:5]")
```

`%v%` *Vertically Add Heatmaps or Annotations to a Heatmap List*

Description

Vertically Add Heatmaps or Annotations to a Heatmap List

Usage

```
x %v% 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 column annotations.

`x` and `y` can also be `NULL`.

Value

A `HeatmapList-class` object.

Author(s)

Zuguang Gu <z.gu@dkfz.de>

See Also

`+.AdditiveUnit` operator is used for horizontal heatmap list.

Examples

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

Index