1.1 General design

A single heatmap is composed of the heatmap body and the heatmap components. The heatmap body can be split by rows and columns. The heatmap components are titles, dendrograms, matrix names and heatmap annotations, which are put on the four sides of the heamap body. The heatmap components are reordered or split according to the heatmap body.

3 Heatmap Annotations

A simple usage of heatmap annotations

```
column_ha = HeatmapAnnotation(foo1 = runif(10), bar1 = anno_b-
arplot(runif(10)))
row_ha = rowAnnotation(foo2 = runif(10), bar2 = anno_barplot(runif-
(10)))
Heatmap(mat, name = "mat", top_annotation = column_ha, right_ann-
otation = row_ha)
```

Color of annotations

```
library(circlize)
col_fun = colorRamp2(c(0, 5, 10), c("blue", "white", "red"))
ha = HeatmapAnnotation(foo = 1:10, col = list(foo = col_fun))
ha = HeatmapAnnotation(bar = sample(letters[1:3], 10, replace =
TRUE),
col = list(bar = c("a" = "red", "b" = "green", "c" = "blue")))
```

specify more than one vectors

```
ha = HeatmapAnnotation(
foo = 1:10,
bar = sample(letters[1:3], 10, replace = TRUE),
col = list(foo = col_fun,
bar = c("a" = "red", "b" = "green", "c" = "blue")))
```

annotation table by df

```
anno_df = data.frame(foo = 1:10,
bar = sample(letters[1:3], 10, replace = TRUE))
ha = HeatmapAnnotation(df = anno_df,
col = list(foo = col_fun,
bar = c("a" = "red", "b" = "green", "c" = "blue")))
```

anno_simple() makes heatmap-like annotations

```
ha = HeatmapAnnotation(foo = anno_simple(1:10, pch = 1,
pt_gp = gpar(col = "red"), pt_size = unit(1:10, "mm")))
3.4 Block annotation
Heatmap(matrix(rnorm(100), 10), name = "mat",
top_annotation = HeatmapAnnotation(foo = anno_block(gp = gpar(fill
= 2:4))),
column_km = 3)
** other annotation graphics
```

3 Heatmap Annotations (cont)

```
ha = HeatmapAnnotation(foo = anno_points(runif(10))) #3.6 Points
annotation**
ha = HeatmapAnnotation(foo = anno_lines(runif(10))) #3.7 Lines
annotation
ha = HeatmapAnnotation(foo = anno_barplot(1:10)) #3.8 Barplot
annotation
## a barplot annotation which visualizes a proportion matrix
m = matrix(runif(4*10), nc = 4)
m = t(apply(m, 1, function(x) x/sum(x)))
ha = HeatmapAnnotation(foo = anno_barplot(m, gp = gpar(fill = 2:5),
bar_width = 1, height = unit(6, "cm")))
ha = HeatmapAnnotation(foo = anno_boxplot(m, height = unit(4, "-
cm"))) #3.9 Boxplot annotation
ha = rowAnnotation(foo = anno_histogram(m)) # applym on rows
#3.10 Histogram annotation
```

3.18 Multiple annotations

```
ha = HeatmapAnnotation(foo = 1:10,
bar = cbind(1:10, 10:1),
pt = anno_points(1:10),
show_legend = c("bar" = FALSE))
Heatmap(matrix(rnorm(100), 10), name = "mat", top_annotation =
ha)
```

4 A List of Heatmaps

The main feature of ComplexHeatmap package is it supports to concatenate a list of heatmaps and annotations horizontally or vertically so that it makes it possible to visualize the associations from various sources of information.

concatenate heatmaps

```
ht1 = Heatmap(mat1, name = "rnorm")
ht2 = Heatmap(mat2, name = "runif")
ht3 = Heatmap(le, name = "letters")
ht1 + ht2 + ht3
```

4.1 Titles

```
draw(ht_list, row_title = "Three heatmaps, row title", row_title_gp =
gpar(col = "red"),
column_title = "Three heatmaps, column title", column_title_gp =
gpar(fontsize = 16))
```

4.3 Gap between heatmaps

```
draw(ht_list, ht_gap = unit(1, "cm"))
```

4.8 Concatenate only the annotations

```
rowAnnotation(foo = 1:12) +
rowAnnotation(bar = anno_barplot(1:12, width = unit(4, "cm")))
```

4.9 Vertical concatenation

```
ht_list = ht1 %v% ht2 %v% ht3; draw(ht_list)
```

2 A Single Heatmap

2.1 Colors

```
## continueous
library(circlize)
col_fun = colorRamp2(c(-2, 0, 2), c("green", "white", "red"), space = "-RGB")
Heatmap(mat, name = "mat", col = col_fun)
## discrete
discrete_mat = matrix(sample(1:4, 100, replace = TRUE), 10, 10)
colors = structure(1:4, names = c("1", "2", "3", "4")) # black, red, green, blue
Heatmap(discrete_mat, name = "mat", col = colors)
```

2.4 Set row and column orders

```
Heatmap(mat, name = "mat", row_order = order(as.numeric(gsub("row", "", rownames(mat)))),
column_order = order(as.numeric(gsub("column", "", colnames(mat)))))
```

2.7 Split by categorical variables

```
## split by a vector
Heatmap(mat, name = "mat",
row_split = rep(c("A", "B"), 9), column_split = rep(c("C", "D"), 12))
## split by a data frame
Heatmap(mat, name = "mat",
row_split = data.frame(rep(c("A", "B"), 9), rep(c("C", "D"), each = 9)))
## Order of slices
Heatmap(mat, name = "mat",
row_split = factor(rep(LETTERS[1:3], 6), levels = LETTERS[3:1]),
column_split = factor(rep(letters[1:6], 4), levels = letters[6:1]),
cluster_row_slices = FALSE,
cluster_column_slices = FALSE)
```

2.12 Get orders and dendrograms

```
small_mat = mat[1:9, 1:9]; ht1 = Heatmap(small_mat); row_order(ht1)
```

2.13 Subset a heatmap

```
ht = Heatmap(mat, name = "mat")
dim(ht); ht[1:10, 1:10]
```

7 OncoPrint

```
## If the separators are in ;:,|, oncoPrint() automatically spit the alteration strings##
get_type_fun = function(x) strsplit(x, ";")[[1]]
```

layer by layer style by specifying alter_fun as a list

```
col = c(snv = "red", indel = "blue")
oncoPrint(mat,
alter_fun = list(
snv = function(x, y, w, h) grid.rect(x, y, w*0.9, h*0.9,
gp = gpar(fill = col["snv"], col = NA)),
indel = function(x, y, w, h) grid.rect(x, y, w*0.9, h*0.4,
gp = gpar(fill = col["indel"], col = NA))), col = col)
```

7.1.2 grid-by-grid style by specifying alter_fun as a single function

```
oncoPrint(mat,
alter_fun = function(x, y, w, h, v) {
if(v["snv"]) grid.rect(x, y, w*0.9, h*0.9, # v["snv"] is a logical value
gp = gpar(fill = col["snv"], col = NA))
if(v["indel"]) grid.rect(x, y, w*0.9, h*0.4, # v["indel"] is a logical value
gp = gpar(fill = col["indel"], col = NA)) }, col = col)
```

7.1.3 Background

If alter_fun is specified as a list, the order of the elements controls the order of adding graphics. There is a special element called background which defines how to draw background and it should be always put as the first element in the alter_fun list.

7.2 Apply to cBioPortal dataset

```
col = c("HOMDEL" = "blue", "AMP" = "red", "MUT" = "#008000")
alter_fun = list(
background = function(x, y, w, h) {
grid.rect(x, y, w-unit(0.5, "mm"), h-unit(0.5, "mm"),
gp = gpar(fill = "#CCCCCC", col = NA)) },
HOMDEL = function(x, y, w, h) {
grid.rect(x, y, w-unit(0.5, "mm"), h-unit(0.5, "mm"),
gp = gpar(fill = col["HOMDEL"], col = NA)) },
AMP = function(x, y, w, h) {
grid.rect(x, y, w-unit(0.5, "mm"), h-unit(0.5, "mm"),
gp = gpar(fill = col["AMP"], col = NA))},
MUT = function(x, y, w, h) {
grid.rect(x, y, w-unit(0.5, "mm"), h*0.33,
```

7 OncoPrint (cont)

gp = gpar(fill = col["MUT"], col = NA)) })

7.2.1 Remove empty rows and columns

```
oncoPrint(mat,
alter_fun = alter_fun, col = col,
remove_empty_columns = TRUE, remove_empty_rows = TRUE,
column_title = column_title, heatmap_legend_param = heatmap_legend_param)
```

7.2.2 Reorder the oncoPrint

```
sample_order = scan(paste0(system.file("extdata", package = "ComplexHeatmap"),
"/sample_order.txt"), what = "character")
oncoPrint(mat,
alter_fun = alter_fun, col = col,
row_order = 1:nrow(mat), column_order = sample_order,
remove_empty_columns = TRUE, remove_empty_rows = TRUE,
column_title = column_title, heatmap_legend_param = heatmap_legend_param)
```

7.2.3 OncoPrint annotations

```
oncoPrint(mat,
alter_fun = alter_fun, col = col,
top_annotation = HeatmapAnnotation(
column_barplot = anno_oncoprint_barplot("MUT", border = TRUE, # only MUT
height = unit(4, "cm"))),
right_annotation = rowAnnotation(
row_barplot = anno_oncoprint_barplot(c("AMP", "HOMDEL"), # only AMP and HOMDEL
border = TRUE, height = unit(4, "cm"),
axis_param = list(side = "bottom", labels_rot = 90))),
remove_empty_columns = TRUE, remove_empty_rows = TRUE,
column_title = column_title, heatmap_legend_param = heatmap_legend_param)
```

add more annotations in HeatmapAnnotation() or rowAnnotation()

```
oncoPrint(mat,
alter_fun = alter_fun, col = col,
remove_empty_columns = TRUE, remove_empty_rows = TRUE,
top_annotation = HeatmapAnnotation(cbar = anno_oncoprint_barplot(),
foo1 = 1:172,
bar1 = anno_points(1:172)),
left_annotation = rowAnnotation(foo2 = 1:26),
right_annotation = rowAnnotation(bar2 = anno_barplot(1:26)),
```

7 OncoPrint (cont)

```
column_title = column_title, heatmap_legend_param = heatmap_legend_param)
```

split the heatmap list

```
ht_list = oncoPrint(mat,
alter_fun = alter_fun, col = col,
column_title = column_title, heatmap_legend_param = heatmap_legend_param) +
Heatmap(matrix(rnorm(nrow(mat)*10), ncol = 10), name = "expr",
width = unit(4, "cm"))
draw(ht_list, row_split = sample(c("a", "b"), nrow(mat), replace = TRUE))
```

Get the subset of rows and columns

```
rownames(ht@matrix); colnames(ht@matrix)
```

5 Legends

5.1 Continuous legends

```
library(circlize)
col_fun = colorRamp2(c(0, 0.5, 1), c("blue", "white", "red"))
lgd = Legend(col_fun = col_fun, title = "foo")
```

5.2 Discrete legends

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

5.4 Heatmap legends

```
m = matrix(rnorm(100), 10)
Heatmap(m, name = "mat", heatmap_legend_param = list(
at = c(-2, 0, 2),labels = c("low", "zero", "high"),
title = "Some values",legend_height = unit(4, "cm"),
title_position = "lefttop-rot"
))
```

5.4 annotation legends

```
ha = HeatmapAnnotation(foo = runif(10), bar = sample(c("f", "m"), 10, replace = TRUE),
annotation_legend_param = list(
foo = list(title = "Fooooooh",at = c(0, 0.5, 1), labels = c("zero", "median", "one") ),
bar = list( title = "Baaaaaaar",at = c("f", "m"),labels = c("Female", "Male"))))
Heatmap(m, name = "mat", top_annotation = ha)
```

5.6 The side of legends

```
draw(ht_list, heatmap_legend_side = "left", annotation_legend_side = "bottom")
```