Introduction to ComplexHeatmap

BRP seminar

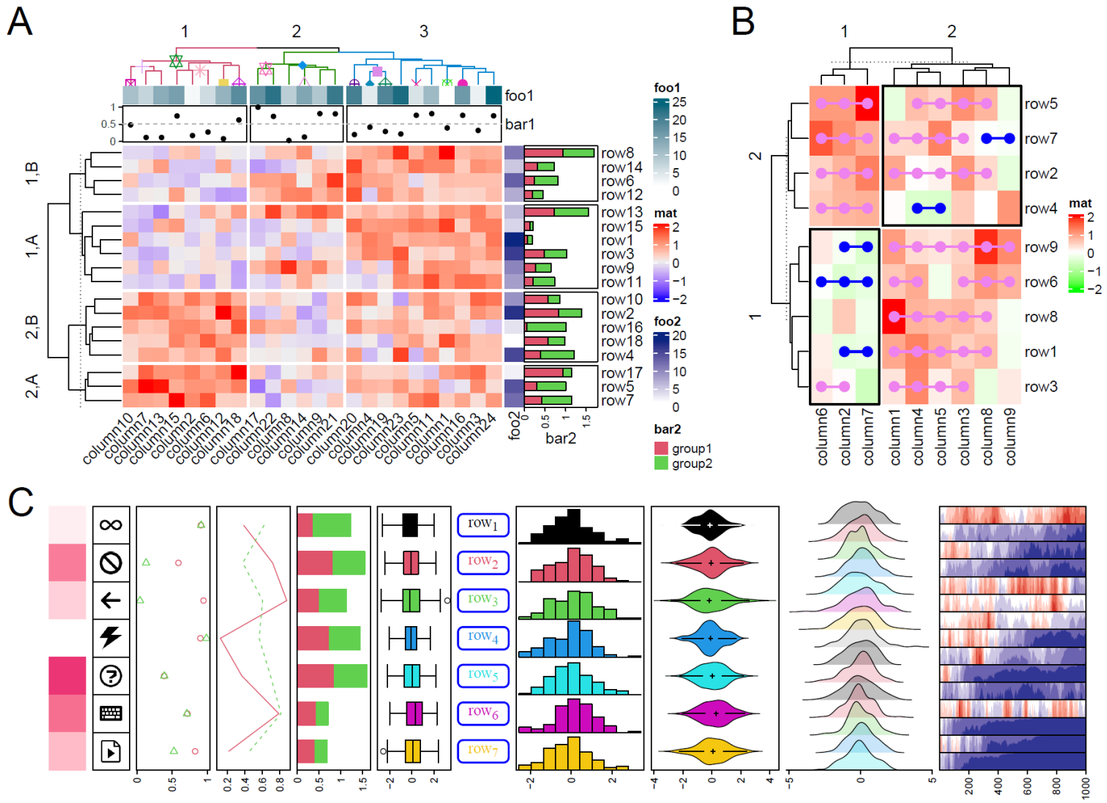
Yuri Kotliarov

November 21, 2023

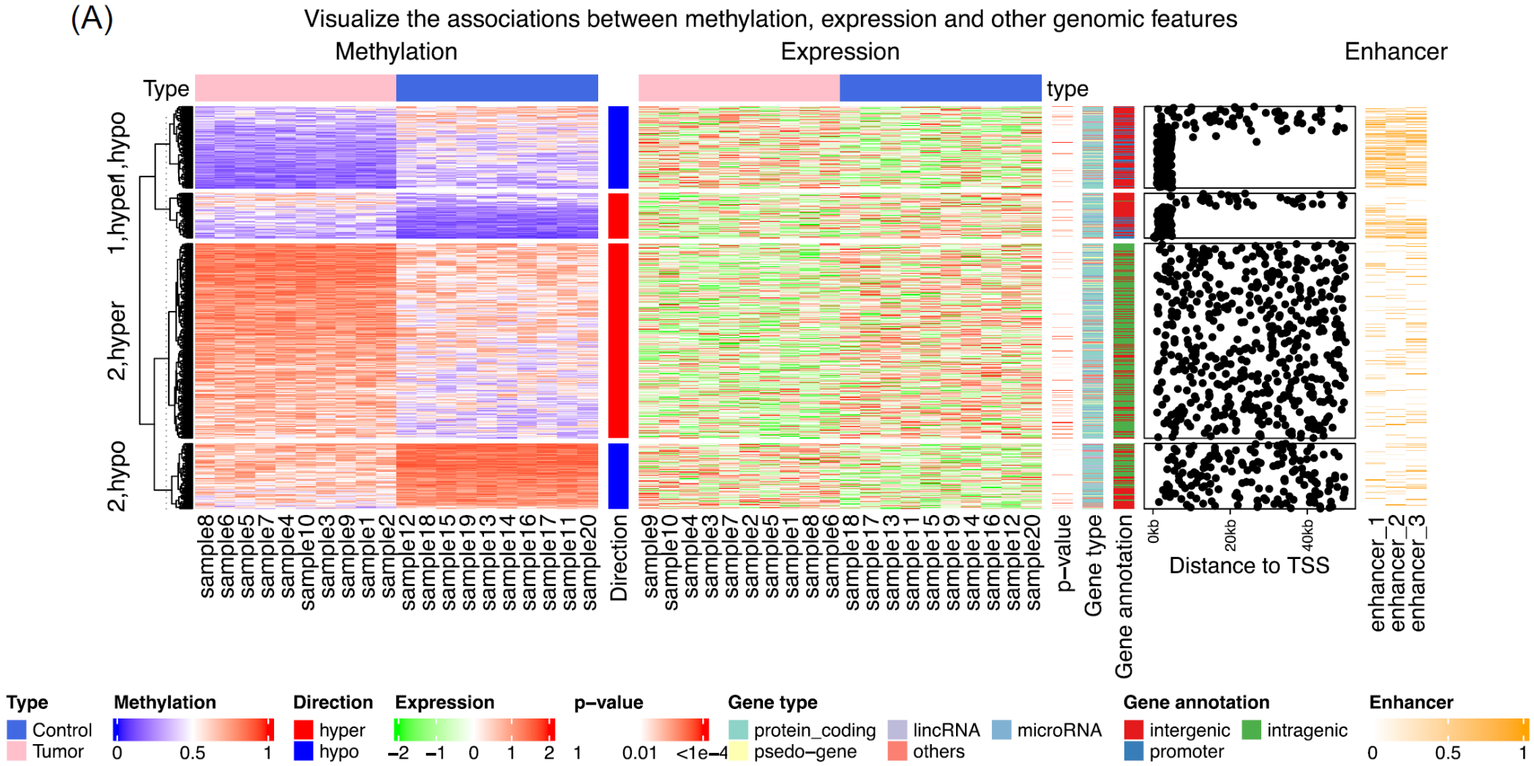
ComplexHeatmap package

- Powerful visualization method for revealing associations between multiple sources of information
- The richest toolset for constructing highly customizable heatmaps
- Modular design
- Automatically concatenates a list of heatmaps with proper row/column ordering
- Automatic legends
- Comprehensive annotations and decorations
- Additional functions and add-on packages for specific visualization tasks

Examples



More Examples



Visualization of the association between DNA methylation, gene expression, and related genomic features

Installation

Install a stable version from Bioconductor:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
if (!requireNamespace("ComplexHeatmap", quietly = TRUE))
  BiocManager::install("ComplexHeatmap")
```

or the most up-to-date version from github:

```
if (!requireNamespace("ComplexHeatmap", quietly = TRUE))
  devtools::install_github("jokergoo/ComplexHeatmap")
```

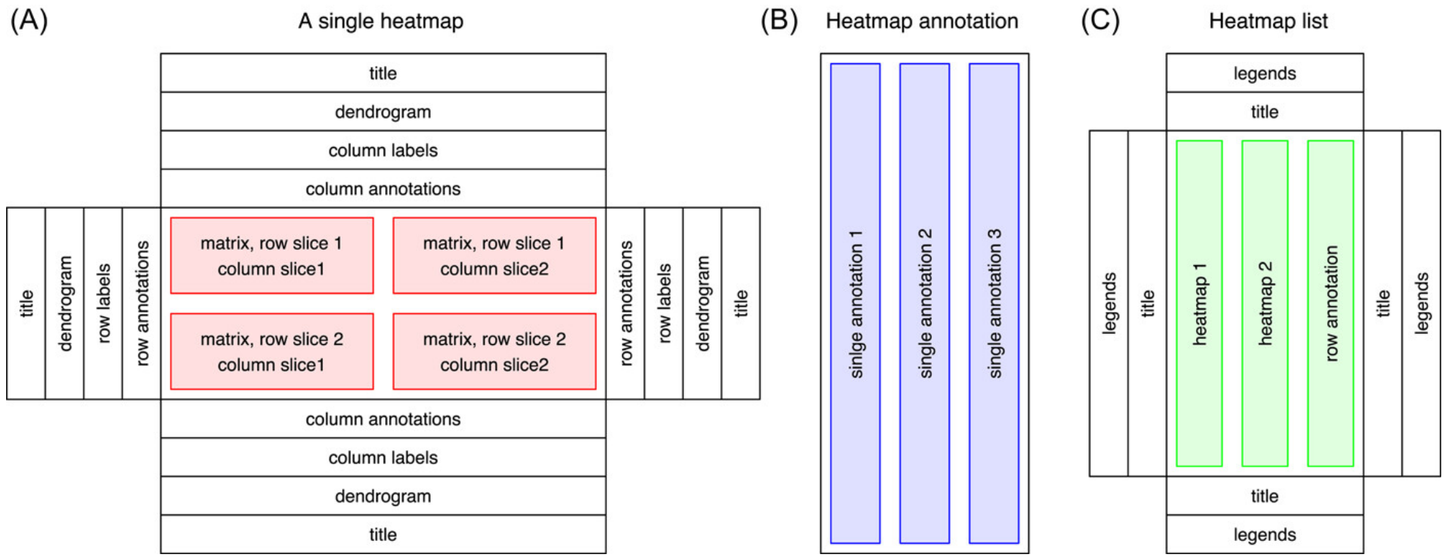
Load the `ComplexHeatmap` package:

```
library(ComplexHeatmap)
```

Three major classes

- *Heatmap* - defines a complete heatmap with multiple components
- *HeatmapAnnotation* - defines a list of annotations with specific graphics
- *HeatmapList* - manages a list of heatmaps and heatmap annotations. It automatically adjusts the correspondence of rows or columns in multiple heatmaps and annotations

Modular design



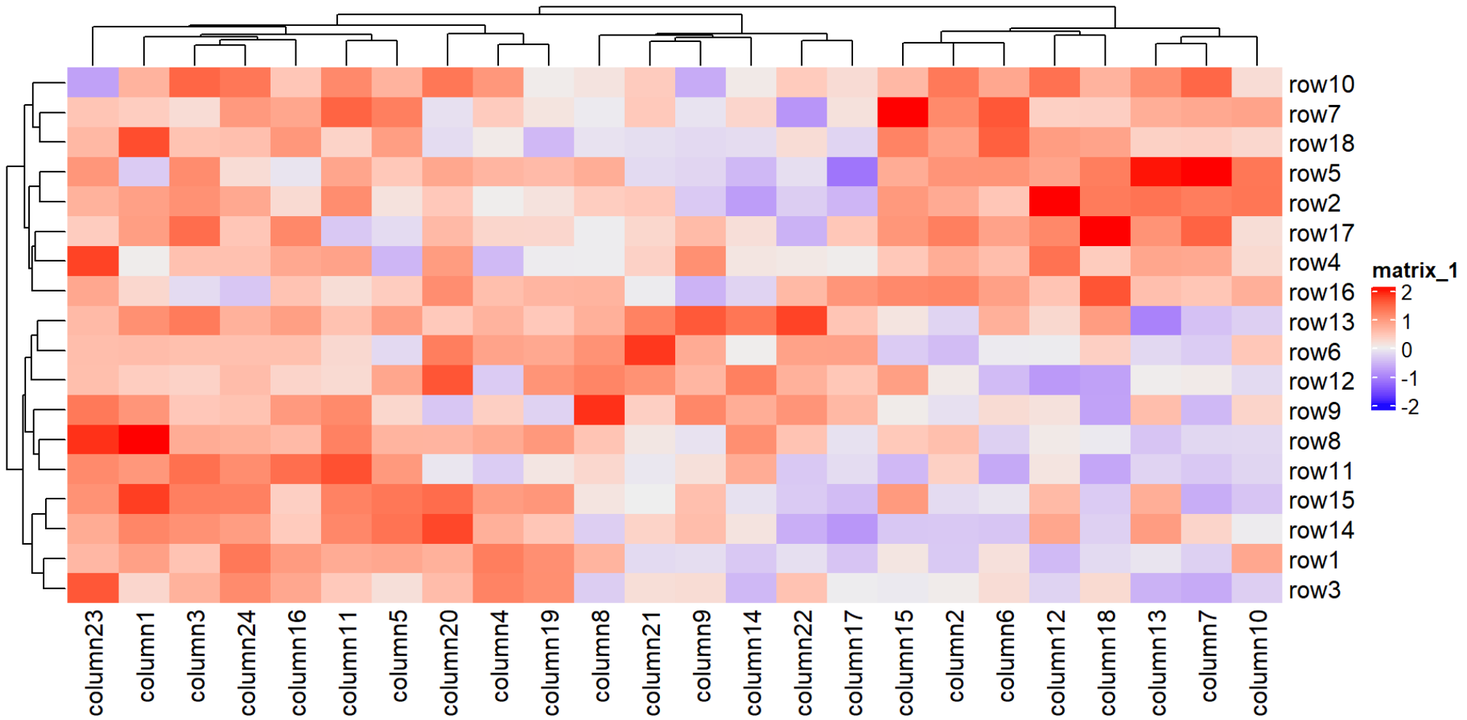
Generate input matrix with random data

```
set.seed(123)
nr1 = 4; nr2 = 8; nr3 = 6; nr = nr1 + nr2 + nr3
nc1 = 6; nc2 = 8; nc3 = 10; nc = nc1 + nc2 + nc3
mat = cbind(rbind(matrix(rnorm(nr1*nc1, mean = 1,   sd = 0.5), nrow = nr1),
                  matrix(rnorm(nr2*nc1, mean = 0,   sd = 0.5), nrow = nr2),
                  matrix(rnorm(nr3*nc1, mean = 0,   sd = 0.5), nrow = nr3)),
            rbind(matrix(rnorm(nr1*nc2, mean = 0,   sd = 0.5), nrow = nr1),
                  matrix(rnorm(nr2*nc2, mean = 1,   sd = 0.5), nrow = nr2),
                  matrix(rnorm(nr3*nc2, mean = 0,   sd = 0.5), nrow = nr3)),
            rbind(matrix(rnorm(nr1*nc3, mean = 0.5, sd = 0.5), nrow = nr1),
                  matrix(rnorm(nr2*nc3, mean = 0.5, sd = 0.5), nrow = nr2),
                  matrix(rnorm(nr3*nc3, mean = 1,   sd = 0.5), nrow = nr3))
            )
mat = mat[sample(nr, nr), sample(nc, nc)] # random shuffle rows and columns
rownames(mat) = paste0("row", seq_len(nr))
colnames(mat) = paste0("column", seq_len(nc))
dim(mat)
```

```
[1] 18 24
```

A Single Heatmap

```
1 Heatmap(mat)
```

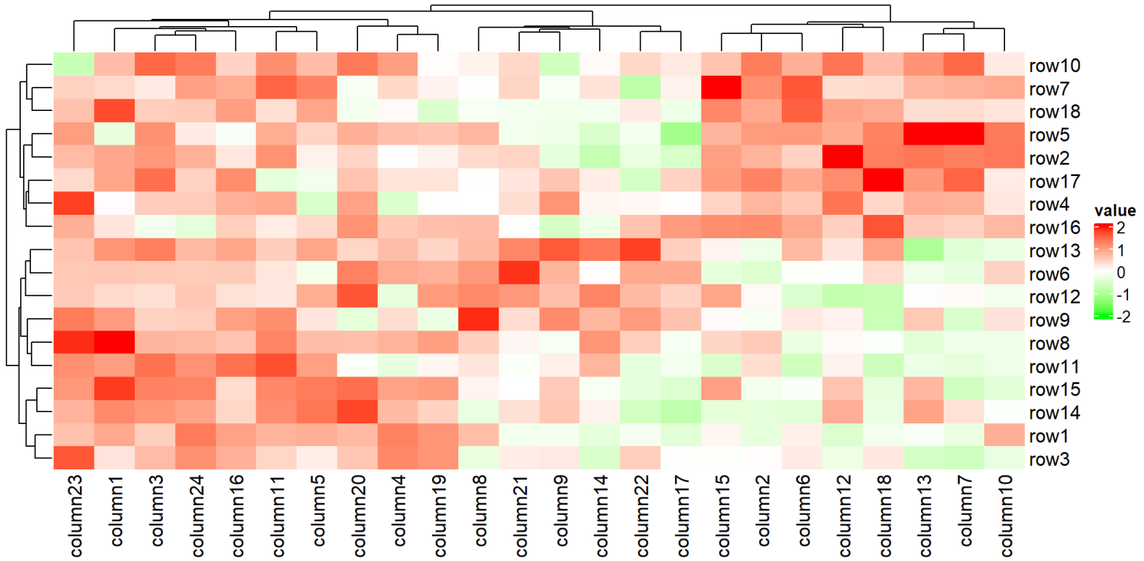


Changing heatmap colors

```
1 library(circlize)
2 col_fun = colorRamp2(c(-2, 0, 2), c("green", "white", "red"))
3 col_fun(seq(-3, 3))
```

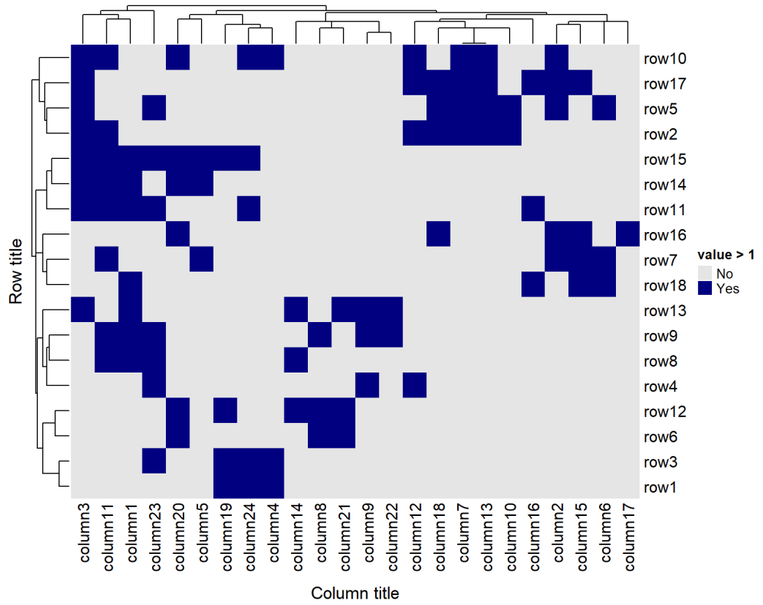
```
[1] "#00FF00FF" "#00FF00FF" "#B1FF9AFF" "#FFFFFFFF" "#FF9E81FF" "#FF0000FF"
[7] "#FF0000FF"
```

```
1 Heatmap(mat, name = "value", col = col_fun)
```



Binary matrix

```
binary_mat = +(mat > 1)

colors = c("0" = "grey90",
           "1" = "navy")

hm = Heatmap(binary_mat,
        col = colors,
        row_title = "Row title",
        row_title_side = "left",
        column_title = "Column title",
        column_title_side = "bottom",
        heatmap_legend_param = list(
                title = "value > 1", at = 0:1,
                labels = c("No", "Yes")))
draw(hm)
```




```
row_order(hm)
```

```
 [1] 10 17  5  2 15 14 11 16  7 18 13  9  8  4 12  6  3  1
```

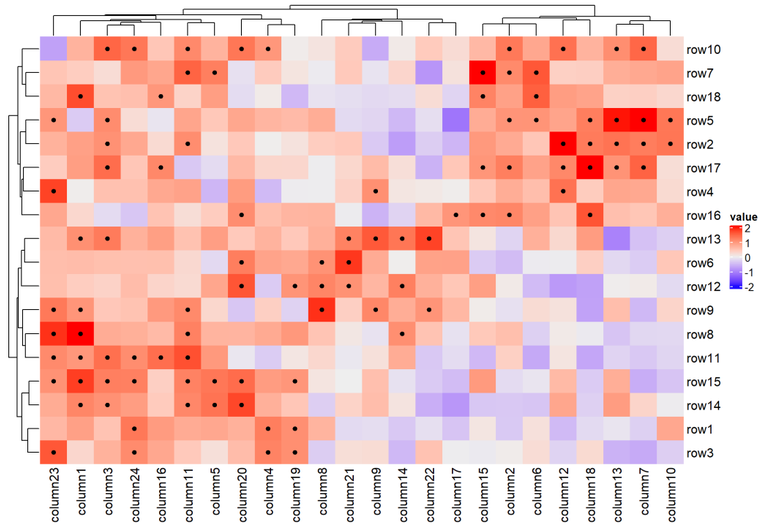
```
column_order(hm)
```

```
 [1]  3 11  1 23 20  5 19 24  4 14  8 21  9 22 12 18  7 13
10 16  2 15  6 17
```

Combining Heatmaps with binary data

We can customize the heatmap cells with *cell_fun* argument. Let's indicate values larger than 1.

```
Heatmap(mat, name = "value",
  cell_fun = function(j, i, x, y,
                      width, height, fill) {
               if (mat[i, j] > 1 & !is.na(mat[i, j]))
                  grid.points(x, y, pch = 20,
                              size = unit(3, "mm"))
             }
        )
```

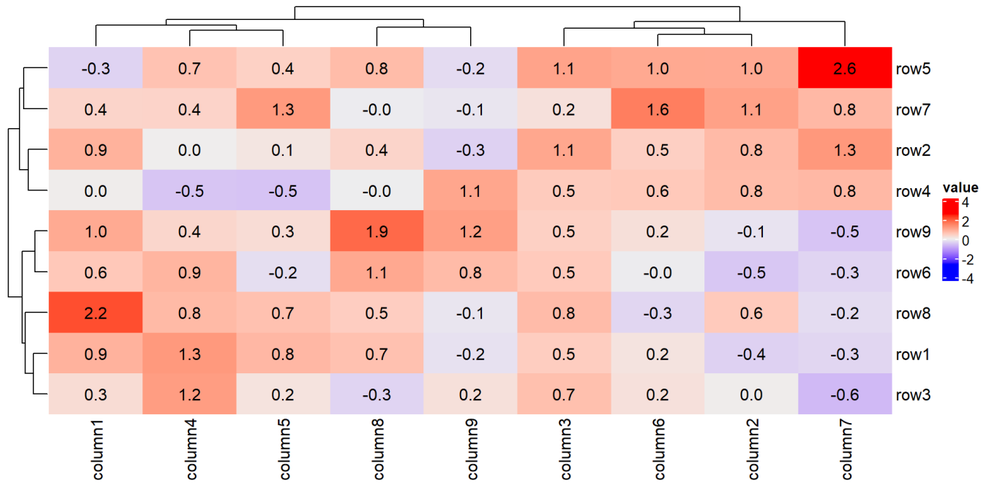


The matrix doesn't need to be the same. For example, we can use matrix of p-values to show values with there significance (even with multiple levels - *, **, ***).

Show values on the heatmap

The similar technique can be used to visulize matrix values (for a relatively small matrix)

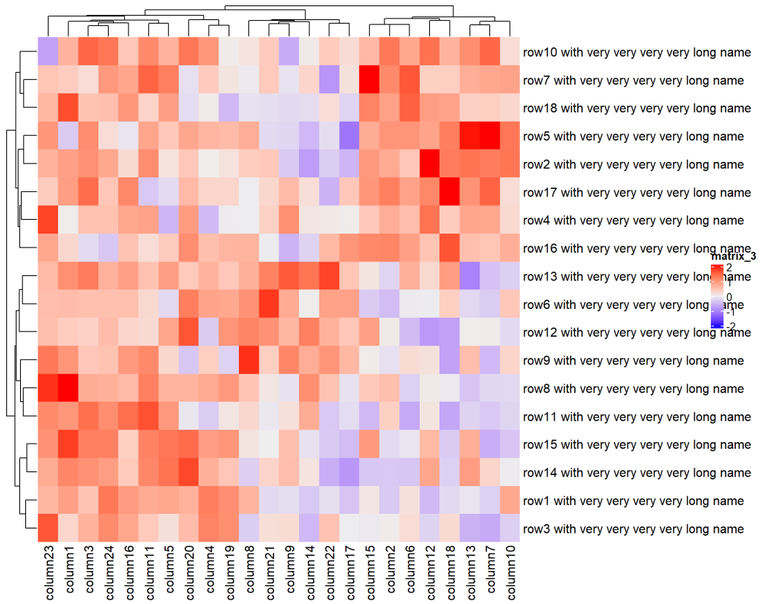
```
small_mat = mat[1:9, 1:9]
Heatmap(small_mat, name = "value",
        cell_fun = function(j, i, x, y, width, height, fill) {
               grid.text(sprintf("%.1f", small_mat[i, j]), x, y,
                         gp = gpar(fontsize = 12)) }
        )
```



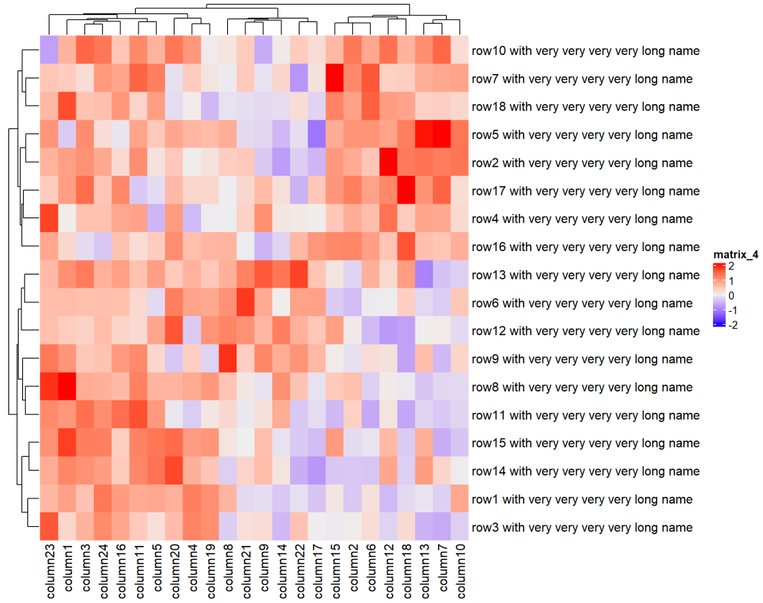
See also *layer_fun* argument for additional heatmap customization techniques.

Hint: long row/column names

```
mat2 = mat
rownames(mat2) = paste(rownames(mat),
                       "with very very very very long name")
Heatmap(mat2)
```



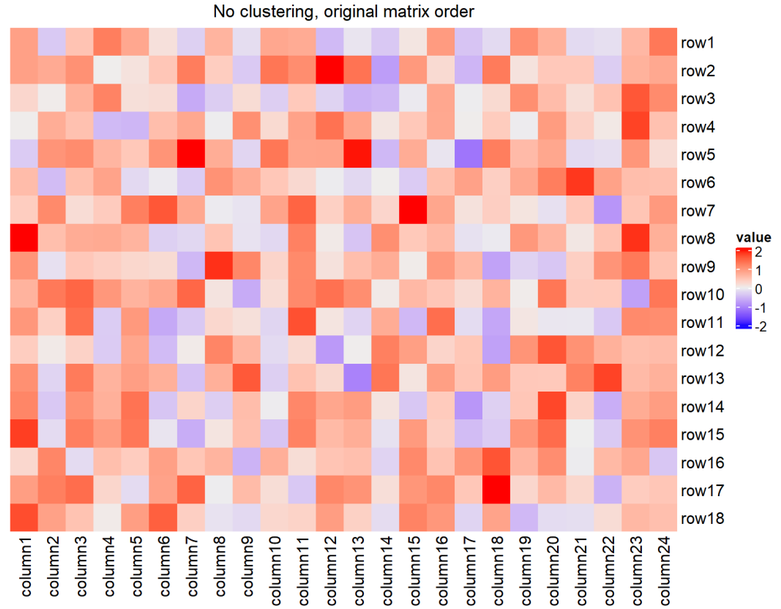

```
Heatmap(mat2,
        row_names_max_width =
          max_text_width(rownames(mat2)))
```



Row and column ordering (without clustering)

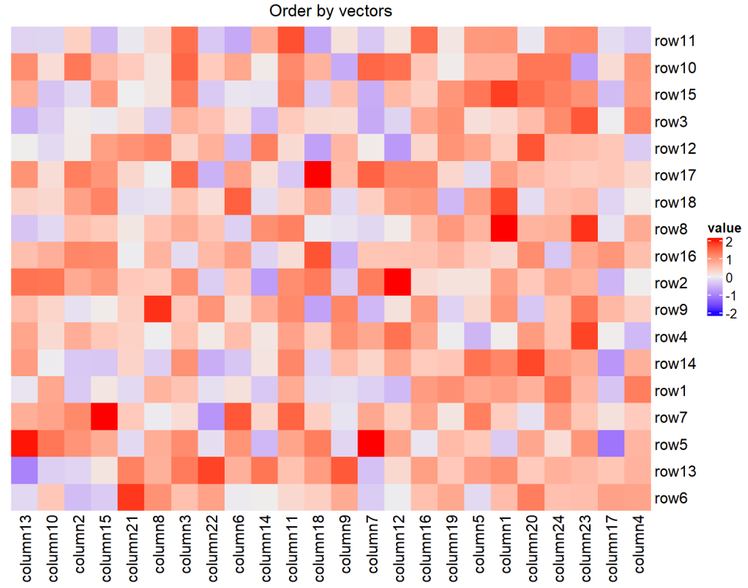
No clustering (original order)

```
Heatmap(mat,
  name = "value",
  cluster_rows = F,
  cluster_columns = F,
  column_title = "No clustering, original matrix order")
```



Order by vectors

```
row_vec = sample(1:nrow(mat))
column_vec = sample(1:ncol(mat))
Heatmap(mat, name = "value",
  row_order = row_vec, column_order = column_vec,
  column_title = "Order by vectors")
```

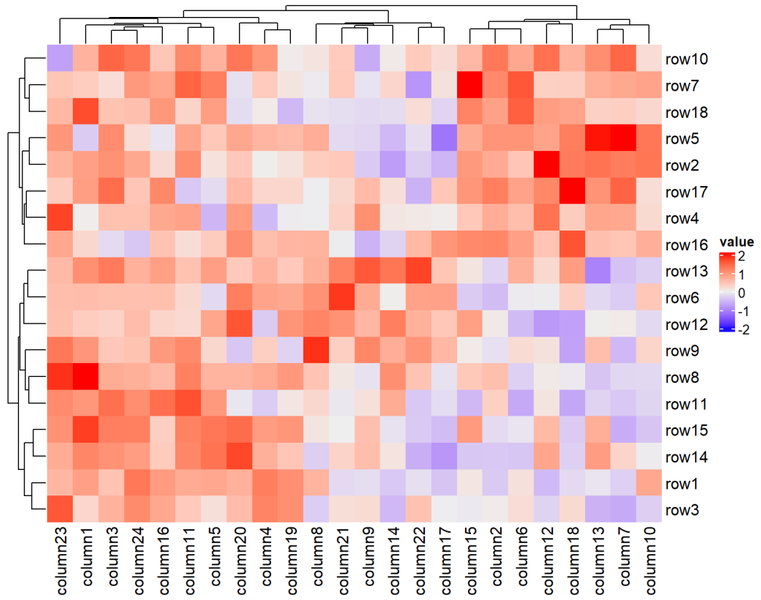


Clustering and splitting

You can specify clustering distance (*clustering_distance_X*) and linkage method (*clustering_method_X*), dendrogramm reordering function (*X_dend_reorder*) as well as a different clustering function (*cluster_X*).

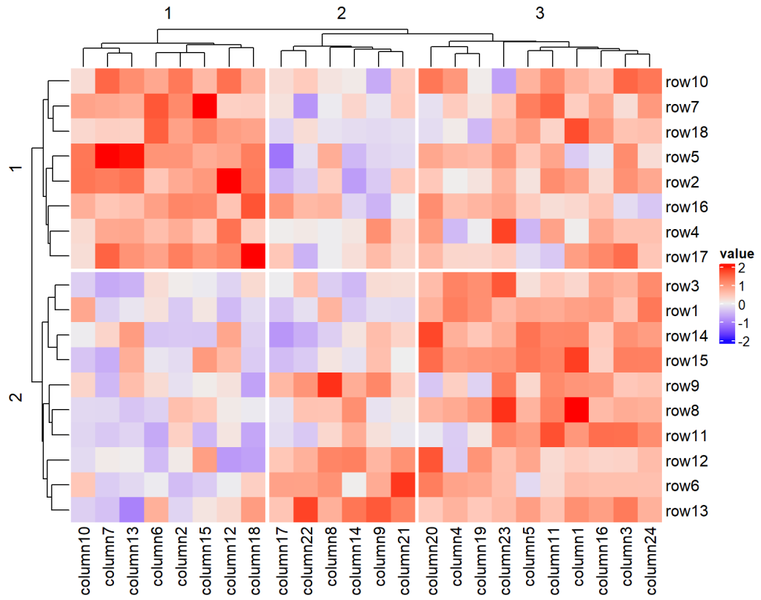
Default clustering

```
Heatmap(mat, name = "value",
        cluster_rows = T, cluster_columns = T)
```



Split dendrogramm branches

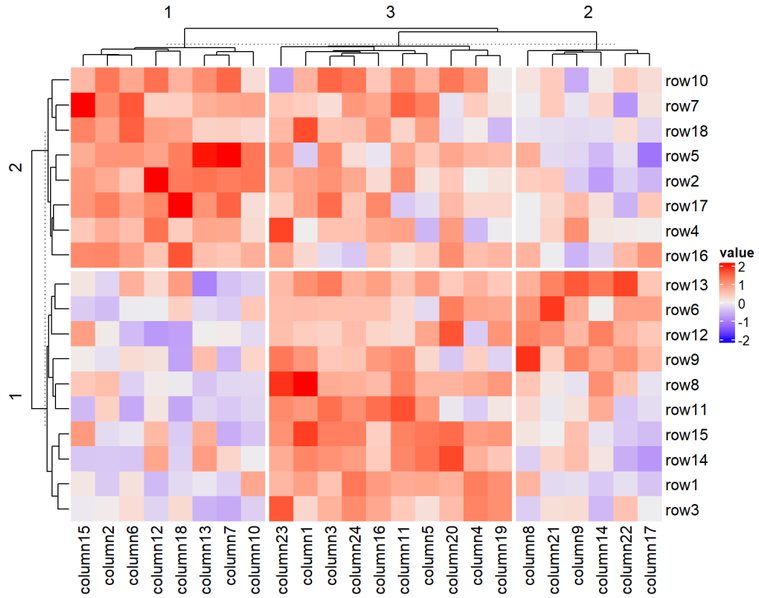
```
Heatmap(mat, name = "value",
        row_split = 2, column_split = 3)
```



Splitting the Heatmap

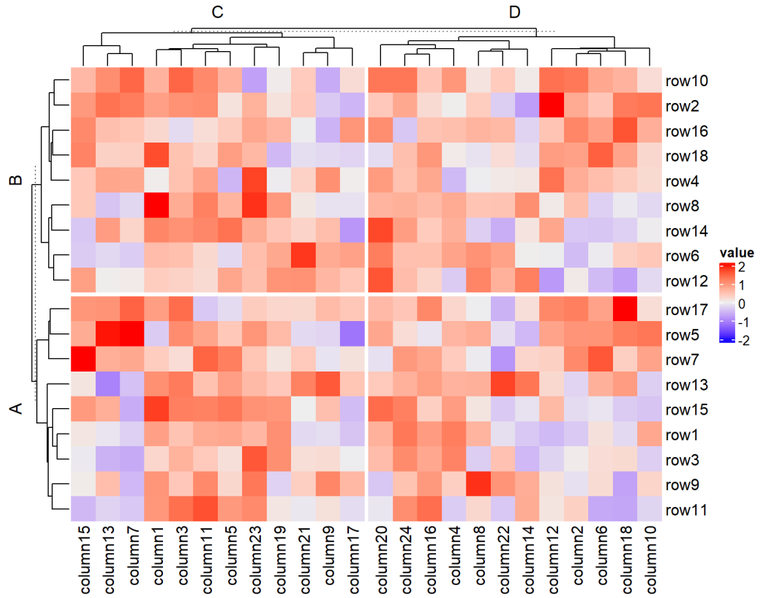
K-mean clustering

```
Heatmap(mat,
        name = "value",
        row_km = 2,
        column_km = 3)
```



Categorical vector(s)

```
row_group = rep(c("A", "B"), 9)
column_group = rep(c("C", "D"), 12)
Heatmap(mat, name = "value",
  row_split = row_group, column_split = column_group)
```

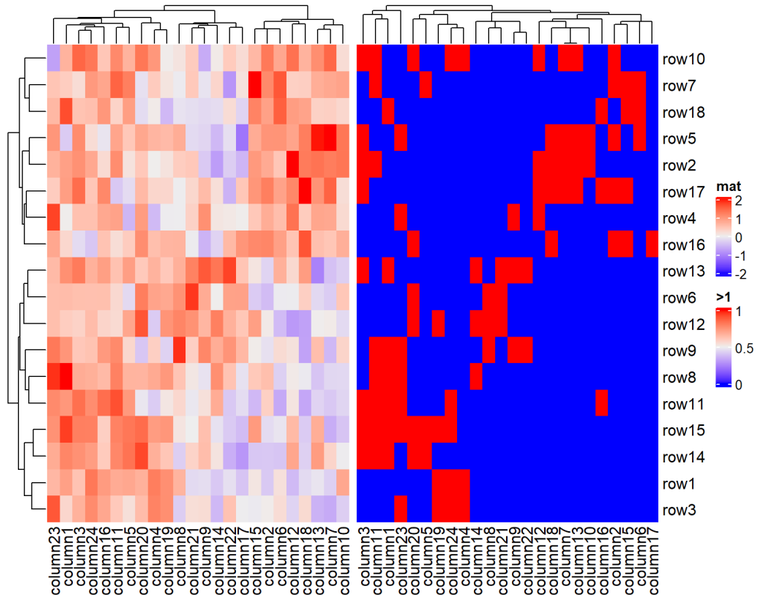


Heatmaps concatenation

Horizontal concatenation

The number of rows of all matrices have to match.

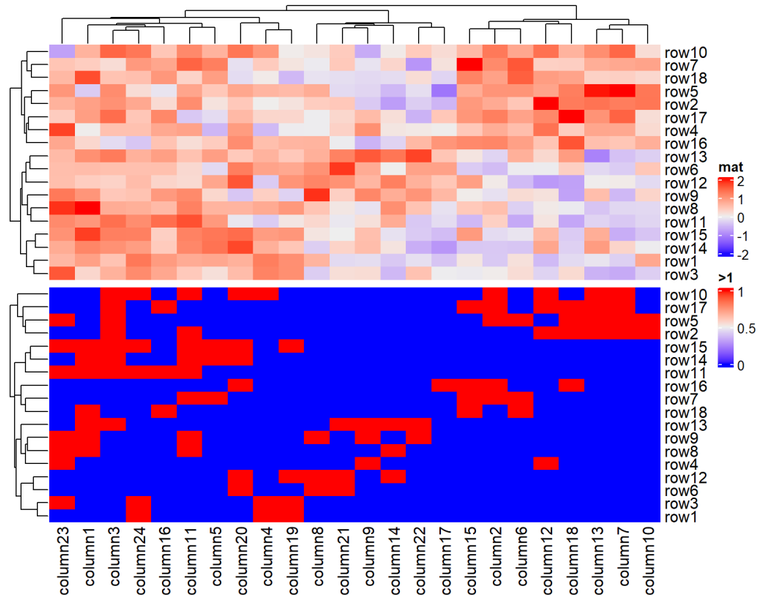
```
1  Heatmap(mat, name = "mat") +
2  Heatmap(binary_mat, name = ">1")
```



Vertical concatenation

The number of columns of all matrices have to match.

```
1  Heatmap(mat, name = "mat") %v%
2  Heatmap(binary_mat, name = ">1")
```

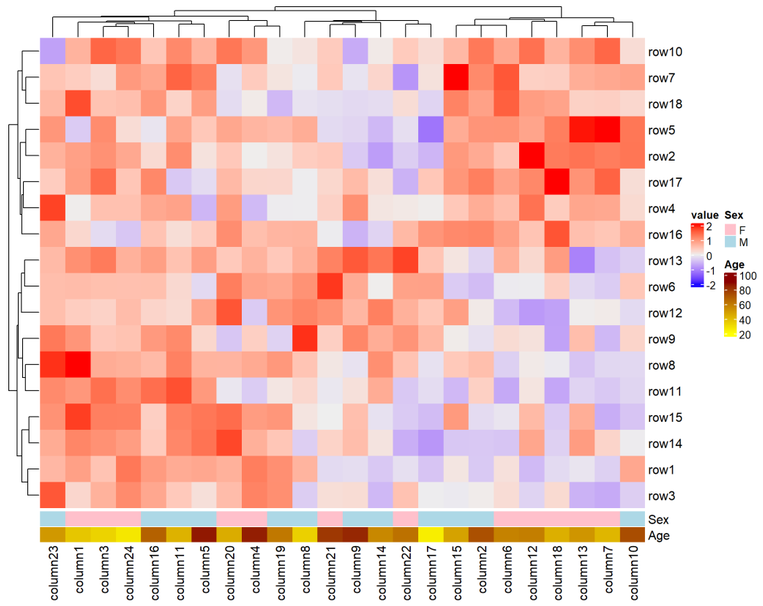


Heatmap annotations

Use *HeatmapAnnotation* function for column annotation, and *rowAnnotation* - for row annotation.

Data frame as data source for color bar annotation

```
df.samp = data.frame(
  Sex = sample(c("M", "F"), 24, replace = T),
  Age = sample(20:90, 24))

col_map = list(
  Sex = c(F = "pink", M = "lightblue"),
  Age = colorRamp2(c(20, 90), c("yellow", "darkred")))

ha = HeatmapAnnotation(df = df.samp, col = col_map)

Heatmap(mat, name = "value",
        bottom_annotation = ha)
```

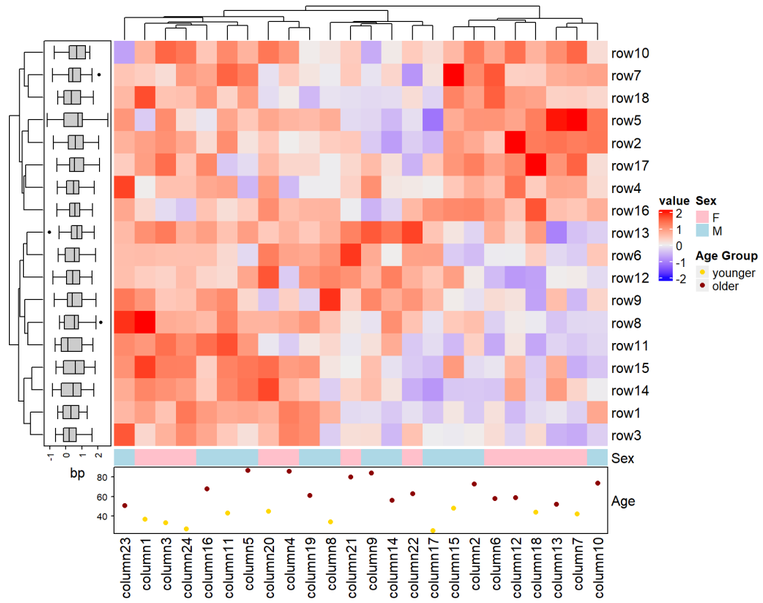


Helper functions and legends

Instead of data frame you can use vectors and matrices (numeric or categorical).

For annotations other than color bar (*anno_simple*) the legends need to be created explicitly, but it's pretty simple.

```
df.samp$Age_group = ifelse(
  df.samp$Age > 50, "older", "younger")
col_sex = c(F = "pink", M = "lightblue")
col_age = c(younger = "gold", older = "darkred")

ha = HeatmapAnnotation(
  Sex = df.samp$Sex, col = list(Sex = col_sex),
  Age = anno_points(df.samp$Age,
          gp = gpar(col = col_age[df.samp$Age_group]),
          height = unit(20,"mm")))
ha_row = rowAnnotation(bp = anno_boxplot(mat))

lgd_list = list(
  Legend(labels = c("younger", "older"),
    title = "Age Group", type = "points", pch = 16,
    legend_gp = gpar(col = col_age)))

hm = Heatmap(mat, name = "value",
```

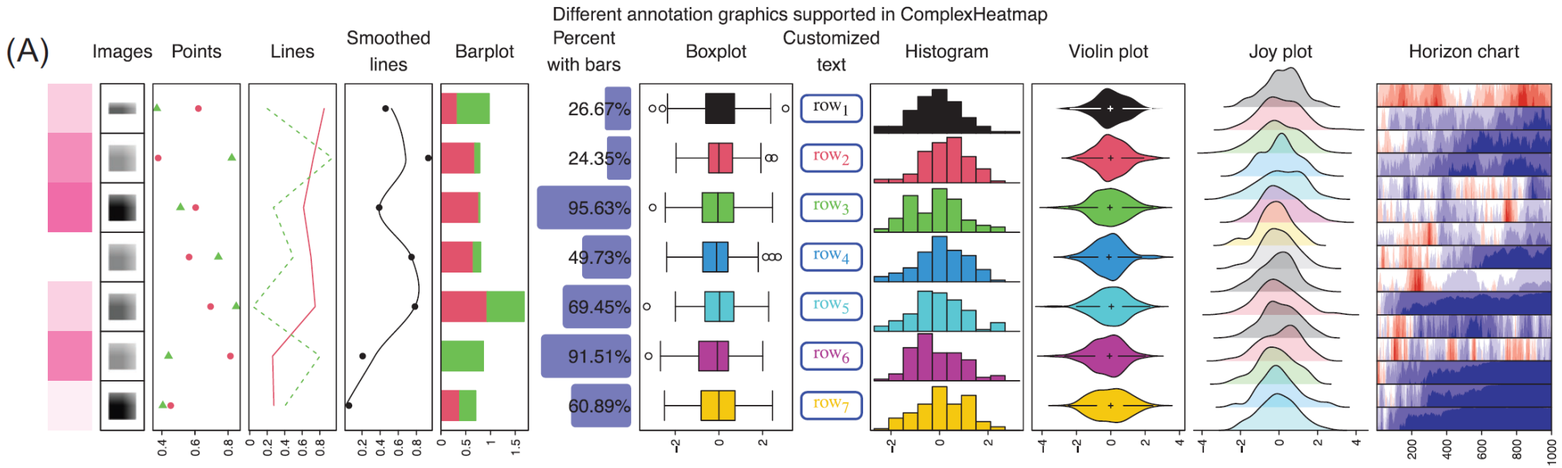


Annotation helper functions

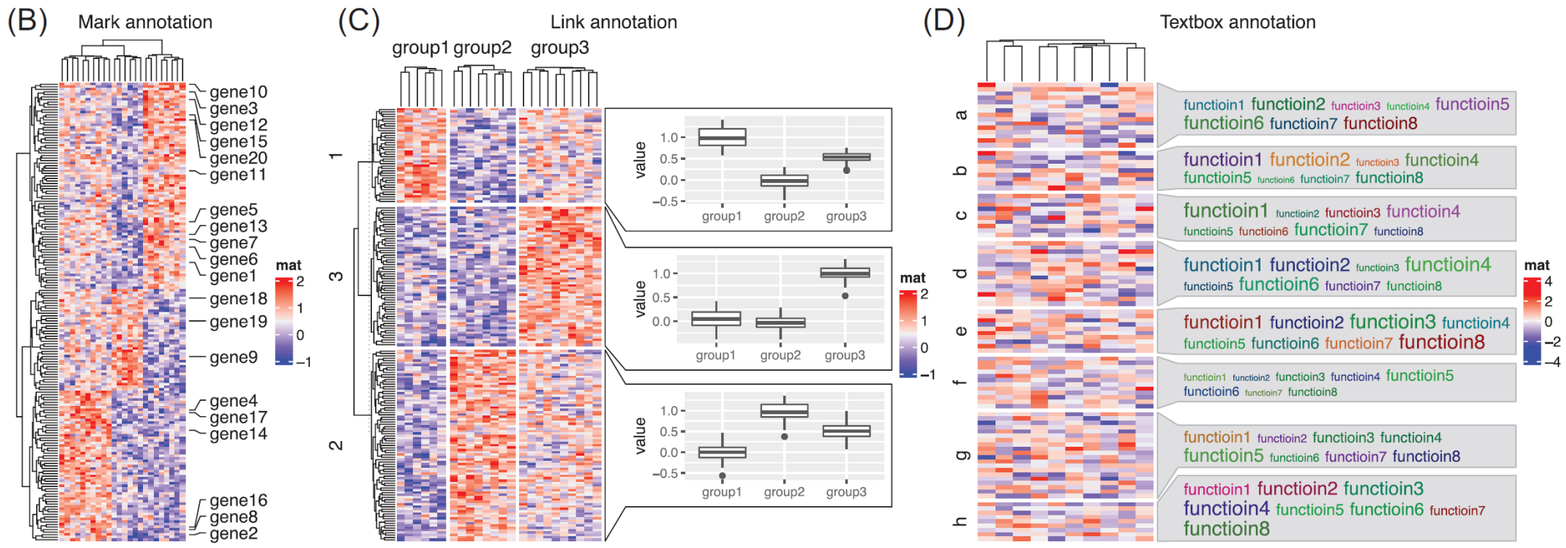
- *anno_simple* (default)
- *anno_points*
- *anno_lines*
- *anno_barplot*
- *anno_numeric*
- *anno_text*
- *anno_textbox*
- *anno_boxplot*
- *anno_histogram*
- *anno_density*
- *anno_joyplot*
- *anno_horizon*
- *anno_summary*
- *anno_image*
- *anno_empty*
- *anno_block*
- *anno_customize*
- *anno_mark*
- *anno_link*
- *anno_zoom*

For more details: https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-annotations.html

Examples of annotation



More advanced annotations



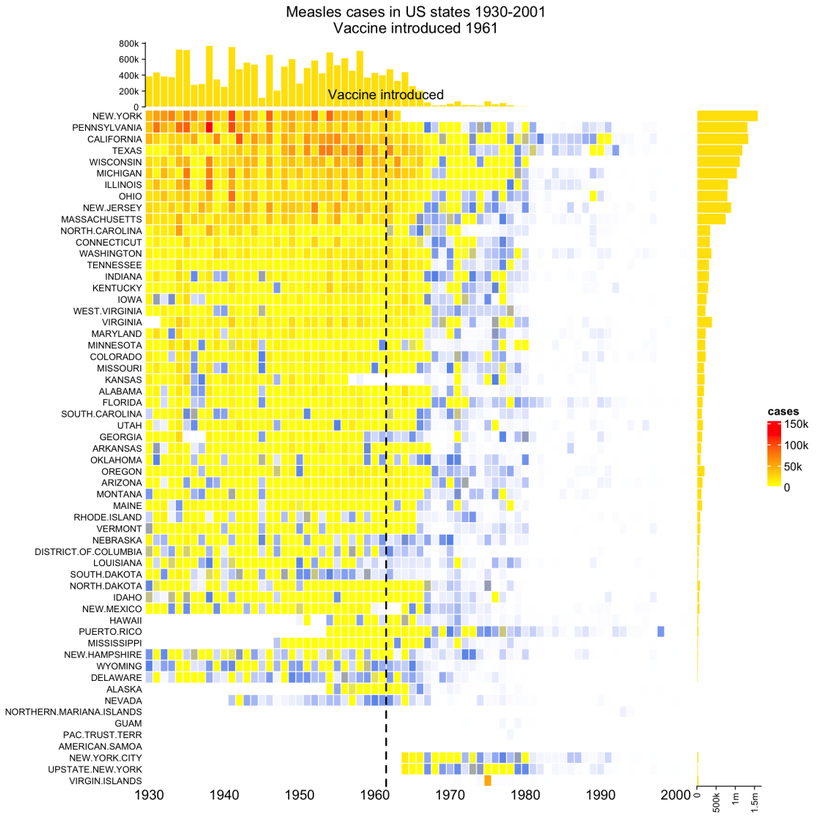
Heatmap Decoration

After Heatmap is created you can still add some decorating elements (text, lines, symbols, etc.) to different parts of the Heatmap (heatmap body, annotations, dendrograms, titles, row and column names).

- *decorate_heatmap_body*
- *decorate_annotation*
- *decorate_dend*
- *decorate_title*
- *decorate_dimnames*
- …

For more information and examples: https://jokergoo.github.io/ComplexHeatmap-reference/book/heatmap-decoration.html

Example: Measles cases in US



Oncoprint - visualizaing genomic alterations

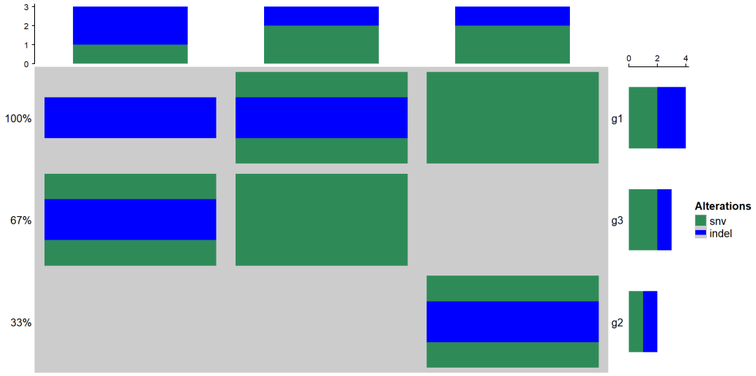
```
mut = read.table(textConnection(
"s1,s2,s3
g1,snv;indel,snv,indel
g2,,snv;indel,
g3,snv,,indel;snv"),
row.names = 1, header = TRUE,
sep = ",", stringsAsFactors = FALSE)

mut = as.matrix(mut)
mut
```

```
   s1          s2          s3
g1 "snv;indel" "snv"       "indel"
g2 ""          "snv;indel" ""
g3 "snv"       ""          "indel;snv"
```

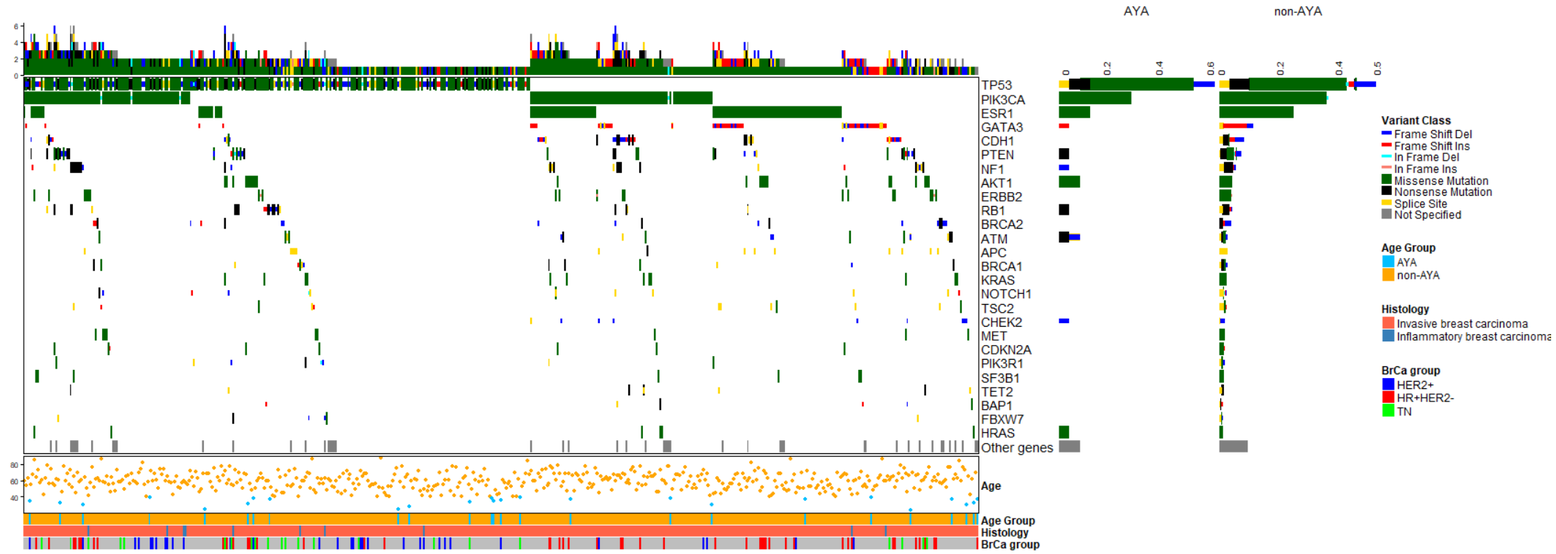
```
get_type_fun = function(x)
  strsplit(x, ";")[[1]]
get_type_fun(mut[1, 1])
```

```
[1] "snv"   "indel"
```

```
col = c(snv = "seagreen", indel = "blue")
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))
    )
oncoPrint(mut, alter_fun = alter_fun, col = col)
```

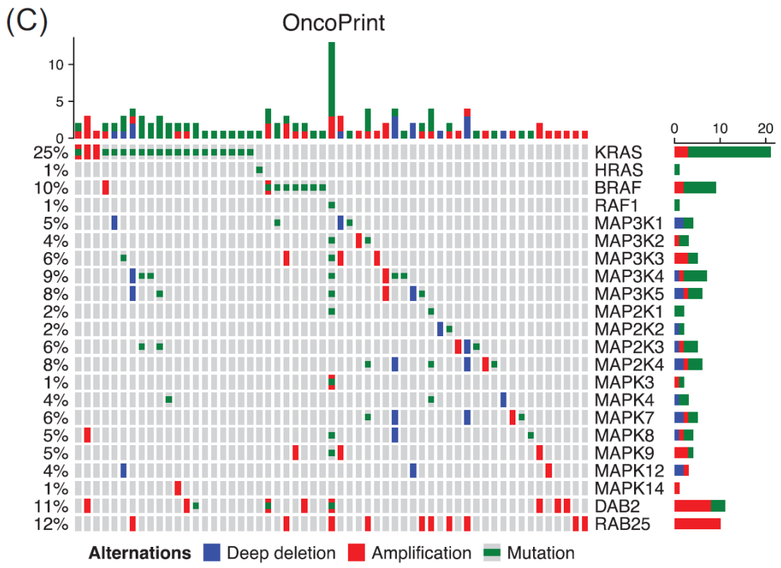


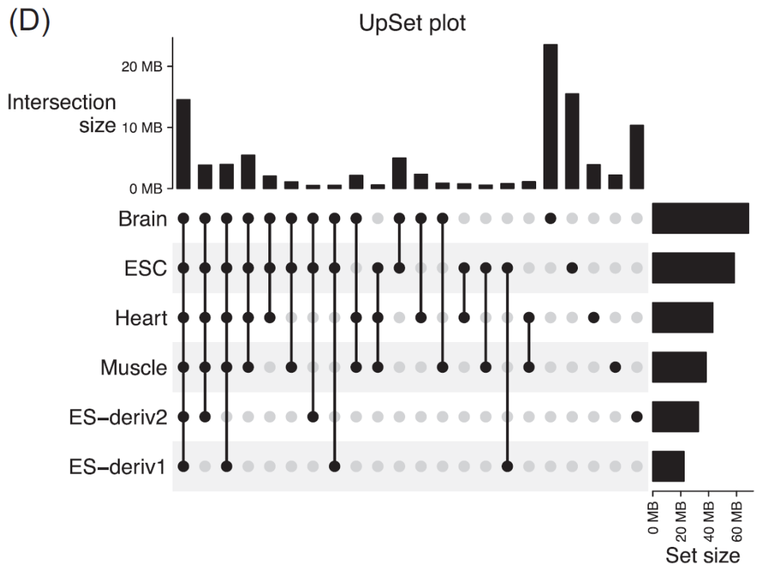
Oncoprint for AYA project



This Oncoprint represents 533 NCI-MATCH Breast cancer patients (including 25 AYA patients below 40 y.o.) and 26 oncogenic genes that were mutated in 6 or more patients.

More ComplexHeatmap plots

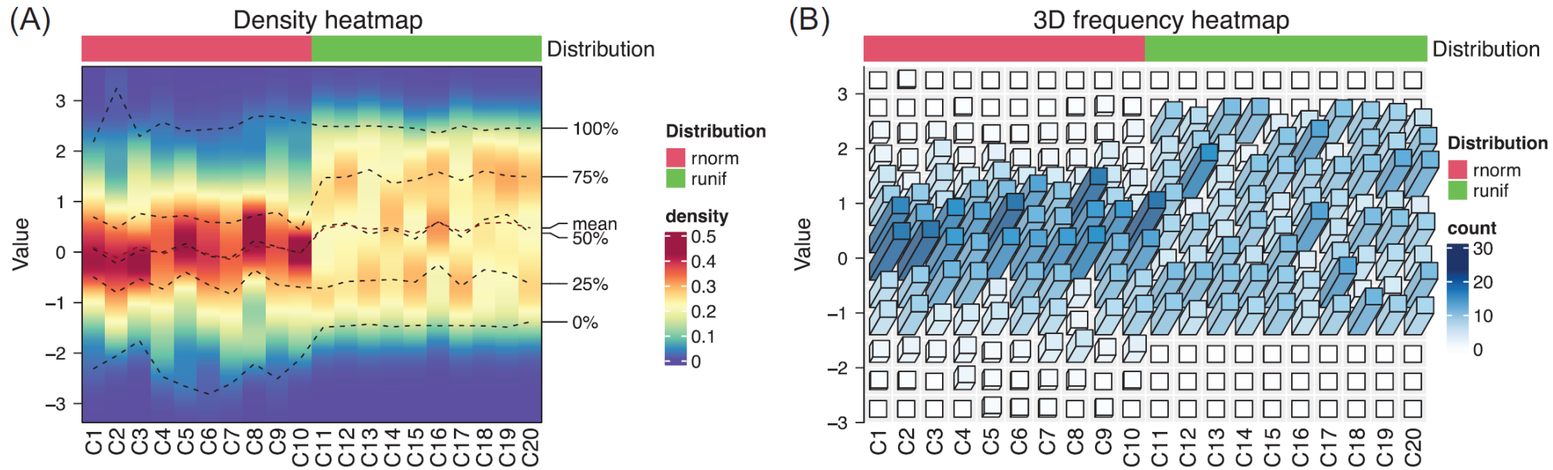




C. The lung adenocarcinoma carcinoma dataset from cBioPortal (a subset)

D. The H3K4me3 ChIP-seq peaks from six human tissues are from the Roadmap project

More ComplexHeatmap plots



Values in the first 10 columns are generated from the normal distribution, and values in the second 10 columns are generated from the uniform distribution.

Related packages

- *InteractiveComplexHeatmap* - converts a Heatmap into a Shiny app
- *simplifyEnrichment* - summarizes gene lists with gene enrichment analysis
- *EnrichedHeatmap* - visualizes the enrichment of a certain type of genomic signal on a list of genomic features of interest. For example, how chromatin modifications are enriched around gene TSSs, or how DNA is lowly methylated around CGIs.

Some Helper packages

- *circlize* - color functions
- *cluster*, *seration*, *biclust* - additional clustering algorithms
- *dendextend* - color and other rendering options for dendrograms
- *dendsort* - dendrogram reordering
- *magick* - rastering Heatmap
- *gridtext* - customized text annotation

Reference and Help

To learn about ComplexHeatmap and related packages see https://jokergoo.github.io/ComplexHeatmap-reference/book.

Github: https://github.com/jokergoo/ComplexHeatmap

Ask questions, report bugs in the issues section of the Github.

Citations

Zuguang Gu, et al., Complex heatmaps reveal patterns and correlations in multidimensional genomic data, Bioinformatics, 2016.

Zuguang Gu. Complex Heatmap Visualization, iMeta, 2022.