How to draw relations

Zuguang Gu <z.gu@dkfz.de>

September 1, 2013

One feature of circular layout is the link (or connector) to represent relations between elements (http://circos.ca/intro/tabular_visualization/). Normally, the relationship can be represented as a matrix in which value in i^{th} row and j^{th} column is kind of strength for the relationship.

We would show how to represent a correlation matrix. First, we need a correlation matrix.

```
> set.seed(12345)
> n = 9
> m = matrix(rnorm(n^2), n, n)
> colnames(m) = letters[1:n]
> m2 = cor(m)
```

Then the range for each element, which is the summation of correlations with other elements.

```
> xlim = cbind(rep(0, n), apply(m2, 2, function(x) sum(abs(x)) - 1))
```

Initialize the layout and add names of elements around each sector. The size of each sector is determined by the range of each element. Here we create the first track by `circos.trackPlotRegion`. In this track, we only want to draw one rectangle and add the name for the element. This is done by `panel.fun` argument to add graphics in the cell as soon as the corresponding cell has been created.

```
> library(circlize)
> factors = rownames(m2)
> colors = 1:n
> par(mar = c(1, 1, 1, 1))
> circos.initialize(factors = factors, xlim = xlim)
> circos.trackPlotRegion(ylim = c(0, 1), factors = factors, bg.border = NA,
+     panel.fun = function(x, y) {
+         xlim = get.cell.meta.data("xlim")
+         current.sector.index = get.cell.meta.data("sector.index")
+         circos.text(mean(xlim), 0.75, labels = current.sector.index,
+             direction = "horizontal")
+         i = get.cell.meta.data("sector.numeric.index")
+         circos.rect(min(xlim), 0, max(xlim), 0.25, col = colors[i])
+     })
```

Finnally we use `circos.link` to draw links from elements to elements, in which red represents positive correlation and green represents negative correlation.

```
> rn = rownames(m2)
> sector.sum = numeric(length(rn))
> for(i in 2:n) {
+     for(j in 1:(i-1)) {
+         sector.index1 = rn[i]
+         sector.index2 = rn[j]
+         circos.link(sector.index1,
+             c(sector.sum[i],sector.sum[i] + abs(m2[i, j])),
+             sector.index2,
+             c(sector.sum[j], sector.sum[j] + abs(m2[i, j])),
+             col = ifelse(m2[i, j] > 0, "#FF0000A0", "#00FF00A0"),
```

```
+                border = "grey")
+            sector.sum[i] = sector.sum[i] + abs(m2[i, j])
+            sector.sum[j] = sector.sum[j] + abs(m2[i, j])
+        }
+ }
> circos.clear()
```

The finnal figure looks like figure 1 (left part) and a more pretty figure is on the right part of figure 1 in which correlatioins are measured by continuous colors.
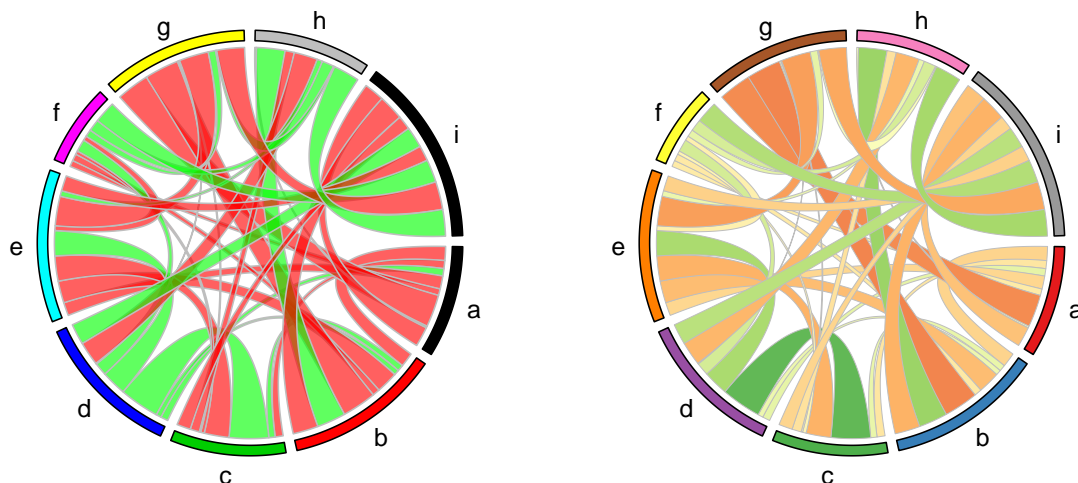

Figure 1: Correlations represented in circular layout

The second example is circular layout for a table in which rows and columns are different measurements. E.g. a table in which rows are countries of Nobel Prize winners and the columns are the areas.

As usual, we first generate an example data. In the example data, letters in upper case is one measurement and letters in lower case is another measurement. The number in the table is the amount of observations in the intersection of two measurements.

```
> mat = matrix(sample(1:100, 18, replace = TRUE), 3, 6)
> rownames(mat) = letters[1:3]
> colnames(mat) = LETTERS[1:6]
> rn = rownames(mat)
> cn = colnames(mat)
```

Construct the `factors` variable and calculate `xlim`. The `factors` are combined with rownames and colnames of the table or matrix.

```
> factors = c(letters[1:3], LETTERS[1:6])
> factors = factor(factors, levels = factors)
> col_sum = apply(mat, 2, sum)
> row_sum = apply(mat, 1, sum)
> xlim = cbind(rep(0, 9), c(row_sum, col_sum))
```

Draw the circular layout of this table (figure 2). In the following code, the sectors with their names in lower case are located in the top half of the circle. This is by specifying `sector.width` in `circos.initialize`. The sum of width of sectors with lower case name is scaled into 1 and so is for sectors with upper case name. As a result, sectors with lower case name will hold half of the circle and sectors of upper case name will hold another half of the circle. We also specify the width of gaps between two

measurements by `gap.degree` in `circos.par`. The colors for different measurements are specified by `bg.col` in `circos.trackPlotRegion`.

```
> par(mar = c(1, 1, 1, 1))
> circos.par(cell.padding = c(0, 0, 0, 0), clock.wise = FALSE,
+     gap.degree = c(2, 2, 10, 2, 2, 2, 2, 2, 10), start.degree = 5)
> circos.initialize(factors = factors, xlim = xlim,
+     sector.width = c(row_sum/sum(row_sum), col_sum/sum(col_sum)))
> circos.trackPlotRegion(factors = factors, ylim = c(0, 1), bg.border = NA,
+     bg.col = c("red", "green", "blue", rep("grey", 6)), track.height = 0.05,
+     panel.fun = function(x, y) {
+         sector.name = get.cell.meta.data("sector.index")
+         xlim = get.cell.meta.data("xlim")
+         circos.text(mean(xlim), 1.5, sector.name, adj = c(0.5, 0))
+
+         if(sector.name %in% rn) {
+             for(i in seq_len(ncol(mat))) {
+                 circos.lines(rep(sum(mat[sector.name, seq_len(i)]), 2), c(0, 1),
+                     col = "white")
+             }
+         } else if(sector.name %in% cn) {
+             for(i in seq_len(nrow(mat))) {
+                 circos.lines(rep(sum(mat[ seq_len(i), sector.name]), 2), c(0, 1),
+                     col = "white")
+             }
+         }
+     })
> col = c("#FF000020", "#00FF0020", "#0000FF20")
> for(i in seq_len(nrow(mat))) {
+     for(j in seq_len(ncol(mat))) {
+         circos.link(rn[i], c(sum(mat[i, seq_len(j-1)]), sum(mat[i, seq_len(j)])),
+                     cn[j], c(sum(mat[seq_len(i-1), j]), sum(mat[seq_len(i), j])),
+                     col = col[i], border = "white")
+     }
+ }
> circos.clear()
```
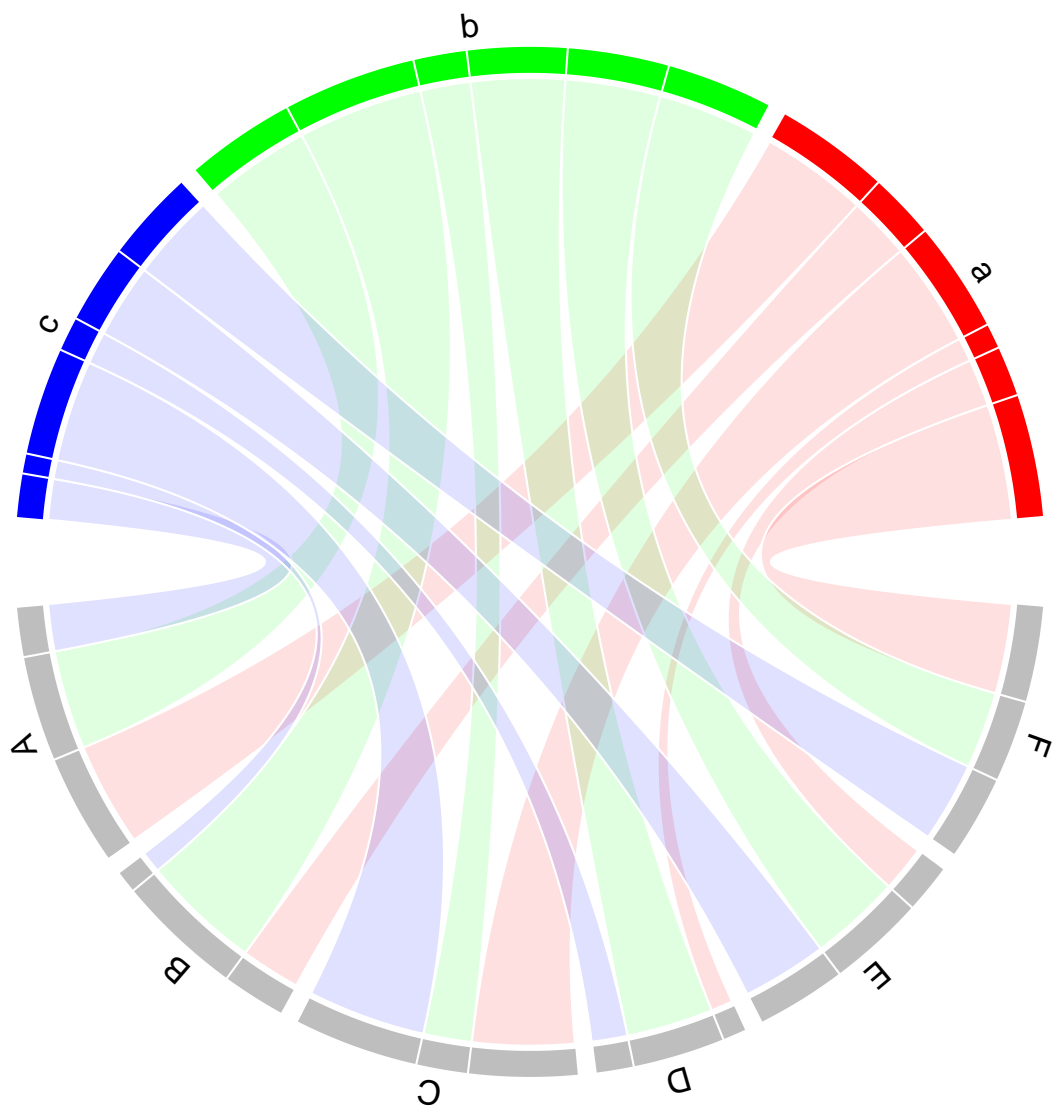
Figure 2: Table in circular layout