

# How to draw a phylogenetic tree

Zuguang Gu <z.gu@dkfz.de>

December 8, 2014

In this short vignette, we will show how to customize the circular style of phylogenetic tree through **circulize** package. Basically, a phylogenetic tree is a dendrogram which is a combination of lines. In R, there are several classes that describe such type of tree such as `hclust`, `dendrogram` and `phylo`. In this example, we will demonstrate how to draw the tree from the `dendrogram` class. Nevertheless, other classes can be converted to `dendrogram` without too much difficulty.

The `bird.orders` data we are using here is from **ape** package. This data set is related to species of birds. We split the tree into six sub trees by `cutree` and convert the data into a `dendrogram` object.

```
library(ape)
data(bird.orders)
hc = as.hclust(bird.orders)
```

```
labels = hc$labels # name of birds
ct = cutree(hc, 6) # cut tree into 6 pieces
n = length(labels) # number of bird species
dgm = as.dendrogram(hc)
```

The `dendrogram` class stores data recursively, which means, a `dendrogram` object contains two children nodes which are also `dendrogram` objects. Thus, we can implement `dendrogram` tree in a recursive way as well.

```
dgm

## 'dendrogram' with 2 branches and 23 members total, at height 28

attributes(dgm)

## $members
## [1] 23
##
## $midpoint
## [1] 4.286621
##
## $height
## [1] 28
##
## $class
## [1] "dendrogram"

length(dgm)

## [1] 2

dgm[[1]]

## 'dendrogram' with 2 branches and 5 members total, at height 25.9

dgm[[2]]

## 'dendrogram' with 2 branches and 18 members total, at height 27
```

For each node in the dendrogram, there are several attributes which are members, midpoint, height, and if the node is the leaf of the tree (i.e. the end of the tree), there would be a binary attribute called leaf. Here what should be noted is that midpoint is the distance to its two children nodes while not the coordinate of the point.

As we mentioned in the main vignette, the x-value for the phylogenetic tree is in fact index. Thus, the x-lim is just the minimum and maximum index of labels in the tree. Since there is only one phylogenetic tree, we only need one "big" sector. In the first track, we plot the name of each bird, with different colors to represent different sub trees.

maxy is the maximum height of tree and is used to calculate the height of the track.

```
library(circlize)
par(mar = c(1, 1, 1, 1))
circos.par(cell.padding = c(0, 0, 0, 0))
circos.initialize(factors = "a", xlim = c(0, n)) # only one sector
maxy = attr(dgm, "height") # maximum height of the trees
circos.trackPlotRegion(ylim = c(0, 1), bg.border = NA, track.height = 0.3,
  panel.fun = function(x, y) {
    for(i in seq_len(n)) {
      circos.text(i-0.5, 0, labels[i], adj = c(0, 0.5),
        facing = "clockwise", niceFacing = TRUE,
        col = ct[labels[i]], cex = 0.7)
    }
  })
```

In the second track, we plot the circular dendrogram. Here we implement the code as `circos.dendrogram` in a recursive way. The `draw.d` function adds lines to its two children nodes, and once it reaches the leaf of the tree, the recursive execution will stop. You can see the advantage of **circlize** that if you replace `circos.lines` to `lines` in this function, then you can almost use the same function to draw dendrogram in the regular Cartesian coordinate system without any error.

Note in following example, argument `maxy` is important if you have more than one trees that are put on a same track.

```
# == param
# -dend a `dendrogram` object
# -maxy the maximum height of the tree is a global attribute,
#      so here it is set as an argument
circos.dendrogram = function(dend, maxy = attr(dend, "height")) {
  labels = as.character(labels(dend))
  x = seq_along(labels) - 0.5 # leaves are places at x = 0.5, 1.5, ..., n - 0.5
  names(x) = labels

  is.leaf = function(object) {
    leaf = attr(object, "leaf")
    if(is.null(leaf)) {
      FALSE
    } else {
      leaf
    }
  }

  draw.d = function(dend, maxy) {
    leaf = attr(dend, "leaf")
    d1 = dend[[1]] # child tree 1
    d2 = dend[[2]] # child tree 2
    height = attr(dend, "height")
    midpoint = attr(dend, "midpoint")

    if(is.leaf(d1)) {
```

```

        x1 = x[as.character(attr(d1, "label"))]
    } else {
        x1 = attr(d1, "midpoint") + x[as.character(labels(d1))[1]]
    }
    y1 = attr(d1, "height")

    if(is.leaf(d2)) {
        x2 = x[as.character(attr(d2, "label"))]
    } else {
        x2 = attr(d2, "midpoint") + x[as.character(labels(d2))[1]]
    }
    y2 = attr(d2, "height")

    # plot the connection line
    circos.lines(c(x1, x1), maxy - c(y1, height), straight = TRUE)
    circos.lines(c(x1, x2), maxy - c(height, height))
    circos.lines(c(x2, x2), maxy - c(y2, height), straight = TRUE)

    # do it recursively
    if(!is.leaf(d1)) {
        draw.d(d1, maxy)
    }
    if(!is.leaf(d2)) {
        draw.d(d2, maxy)
    }
}

draw.d(dend, maxy)
}

```

Finally add the circular dendrogram to the second track:

```

circos.trackPlotRegion(ylim = c(0, maxy), bg.border = NA,
    track.height = 0.4, panel.fun = function(x, y) {
        circos.dendrogram(dgm, maxy)
    })

circos.clear()

```

In figure 1 (top), we cut the dendrogram into six parts and mark the text with different colors. Once you know the basic rule for drawing such circular dendrogram, it would be flexible to customize your figures such as adding bars, lines, points, or even highlighting some parts on the tree.

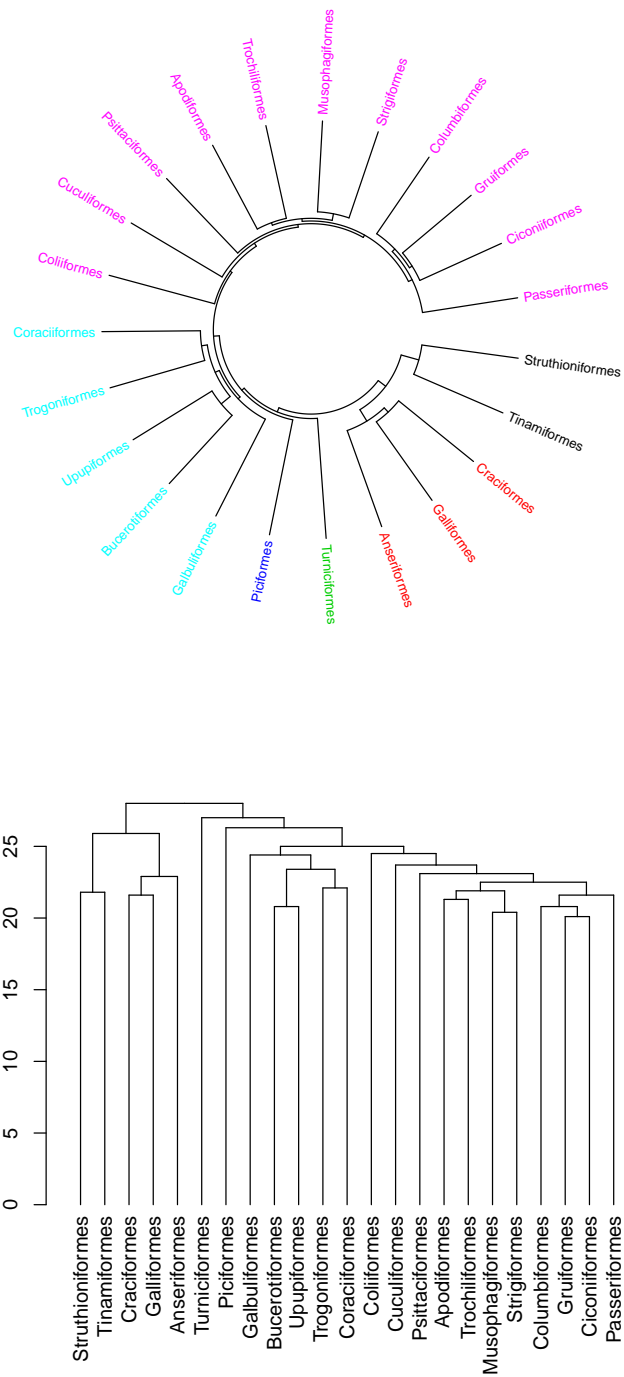


Figure 1: A simple phylogenetic tree. Top: circular style; Bottom: normal style.