

Some interesting graphics

Zuguang Gu <z.gu@dkfz.de>

November 25, 2015

The first example is a clock. The key function here is `circos.axis` (figure 1). The whole circle only contains one sector in which major tick at 0 is overlapping with major tick at 12. The two arrows are plotted in the canvas coordinate. An example of a real-time clock is in *Examples* section in the help page of `circos.axis`.

```
library(circlize)
factors = "a" # any name is OK
circos.par(gap.degree = 0, cell.padding = c(0, 0, 0, 0), start.degree = 90)
circos.initialize(factors = factors, xlim = c(0, 12))
circos.trackPlotRegion(factors = factors, ylim = c(0, 1), bg.border = NA)
circos.axis(sector.index = "a", major.at = 0:12, labels = "",
            direction = "inside", major.tick.percentage = 0.3)
circos.text(1:12, rep(0.5, 12), 1:12, facing = "downward")
arrows(0, 0, 0, 0.7)
arrows(0, 0, 0.4, 0)
```

```
circos.clear()
```

The second example is a dartboard. In the figure, tracks are assigned with different heights and each cell is initialized with different colors (figure 2). The most inside green ring and red circle are plotted by `draw.sector`.

```
factors = 1:20 # just indicate there are 20 sectors
circos.par(gap.degree = 0, cell.padding = c(0, 0, 0, 0),
            start.degree = 360/20/2, track.margin = c(0, 0), clock.wise = FALSE)
circos.initialize(factors = factors, xlim = c(0, 1))
circos.trackPlotRegion(ylim = c(0, 1), factors = factors, bg.col = "black",
                       track.height = 0.15)
circos.trackText(rep(0.5, 20), rep(0.5, 20),
                 labels = c(13, 4, 18, 1, 20, 5, 12, 9, 14, 11, 8, 16, 7, 19, 3, 17, 2, 15, 10, 6),
                 factors = factors, col = "#EEEEEE", font = 2, facing = "downward")
circos.trackPlotRegion(ylim = c(0, 1), factors = factors,
                       bg.col = rep(c("#E41A1C", "#4DAF4A"), 10), bg.border = "#EEEEEE", track.height = 0.05)
circos.trackPlotRegion(ylim = c(0, 1), factors = factors,
                       bg.col = rep(c("black", "white"), 10), bg.border = "#EEEEEE", track.height = 0.275)
circos.trackPlotRegion(ylim = c(0, 1), factors = factors,
                       bg.col = rep(c("#E41A1C", "#4DAF4A"), 10), bg.border = "#EEEEEE", track.height = 0.05)
circos.trackPlotRegion(ylim = c(0, 1), factors = factors,
                       bg.col = rep(c("black", "white"), 10), bg.border = "#EEEEEE", track.height = 0.375)
draw.sector(center = c(0, 0), start.degree = 0, end.degree = 360,
            rou1 = 0.1, col = "#4DAF4A", border = "#EEEEEE")
draw.sector(center = c(0, 0), start.degree = 0, end.degree = 360,
            rou1 = 0.05, col = "#E41A1C", border = "#EEEEEE")
```

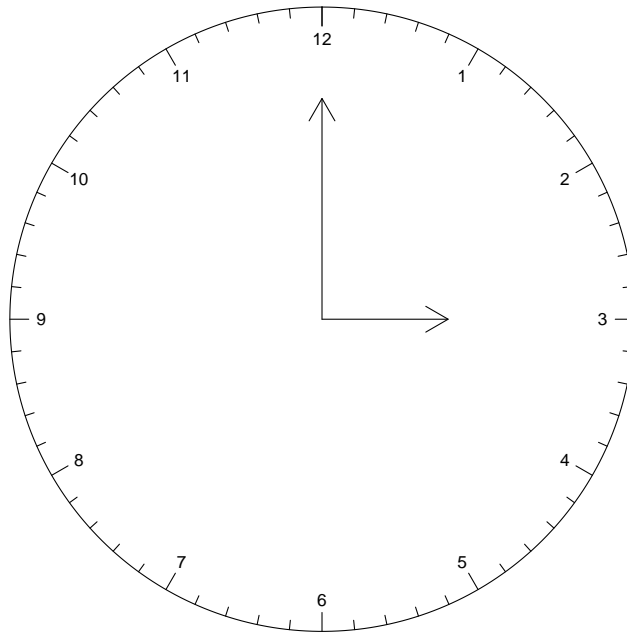


Figure 1: A clock.

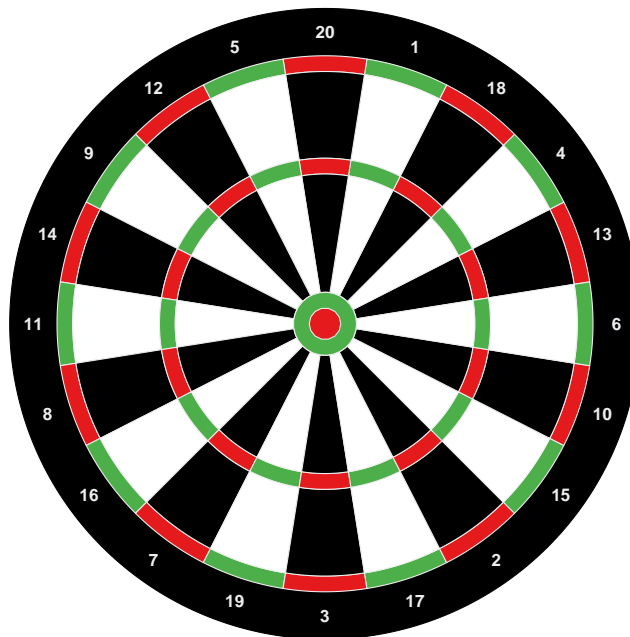


Figure 2: A dartboard.

```
circos.clear()
```

The third example is Bagua (<https://en.wikipedia.org/wiki/Bagua>). The key functions are `circos.rect` and `draw.sector` (figure 3).

Bagua is originated several thousands years ago in China. It is the source of almost all ancient Chinese philosophy. It abstracts the rule of universe into base signs and combination of the two basic signs generates the whole system of the universe.

Inside Bagua, there is the Taiji ([http://en.wikipedia.org/wiki/Taiji_\(philosophy\)](http://en.wikipedia.org/wiki/Taiji_(philosophy))). Taiji refers to the most original state at the creation of the universe. In ancient Chinese philosophy system, at the very beginning, the whole world is a huge mass of air (chaos). Then the lighter air floated up and created sky while heavier sanked down and created ground. The upper world is called Yang and the bottom world is called Ying. And that is Taiji.

So look at Taiji, you can see there are two states interacting with each other. The white one and the black one gradually transformed into each other at the end. And in the center of white and black, the opposite color is generated. In real world, Taiji can represent all phenomenon that is of dualism. Such as male and female, correct and wrong. However things would change, good thing would become bad thing as time goes by, and bad thing also would turn good according how you look at the world.

```
factors = 1:8
circos.par(start.degree = 22.5, gap.degree = 6)
circos.initialize(factors = factors, xlim = c(0, 1))

# yang yao is -
add_yang_yao = function() {
  circos.rect(0,0,1,1, col = "black")
}

# yin yao is --
add_yin_yao = function() {
  circos.rect(0,0,0.45,1, col = "black")
  circos.rect(0.55,0,1,1, col = "black")
}

circos.trackPlotRegion(ylim = c(0, 1), factors = factors, bg.border = NA,
  panel.fun = function(x, y) {
    i = get.cell.meta.data("sector.numeric.index")
    if(i %in% c(2, 5, 7, 8)) add_yang_yao() else add_yin_yao()
  }, track.height = 0.1)

circos.trackPlotRegion(ylim = c(0, 1), factors = factors, bg.border = NA,
  panel.fun = function(x, y) {
    i = get.cell.meta.data("sector.numeric.index")
    if(i %in% c(1, 6, 7, 8)) add_yang_yao() else add_yin_yao()
  }, track.height = 0.1)

circos.trackPlotRegion(ylim = c(0, 1), factors = factors, bg.border = NA,
  panel.fun = function(x, y) {
    i = get.cell.meta.data("sector.numeric.index")
    if(i %in% c(4, 5, 6, 7)) add_yang_yao() else add_yin_yao()
  }, track.height = 0.1)

# the bottom of the most recent track
r = get.cell.meta.data("cell.bottom.radius") - 0.1
# draw taiji, note default order is clock wise for `draw.sector`
draw.sector(center = c(0, 0), start.degree = 90, end.degree = -90,
  roul = r, col = "black", border = "black")
draw.sector(center = c(0, 0), start.degree = 270, end.degree = 90,
  roul = r, col = "white", border = "black")
draw.sector(center = c(0, r/2), start.degree = 0, end.degree = 360,
```



Figure 3: Bagua and Taiji.

```
rou1 = r/2, col = "white", border = "white")
draw.sector(center = c(0, -r/2), start.degree = 0, end.degree = 360,
rou1 = r/2, col = "black", border = "black")
draw.sector(center = c(0, r/2), start.degree = 0, end.degree = 360,
rou1 = r/8, col = "black", border = "black")
draw.sector(center = c(0, -r/2), start.degree = 0, end.degree = 360,
rou1 = r/8, col = "white", border = "white")
```

```
circos.clear()
```

Figure 4 is a circular style of Keith Haring's doodle. The circular transformation is as follows: 1. use `jpeg` package to read RGB information for each pixel in the original figure; 2. use `circos.rect` to draw every pixel into the circle. Source code for generating the figure can be found at <http://jokergoo.github.io/circlize/example/doodle.html>.

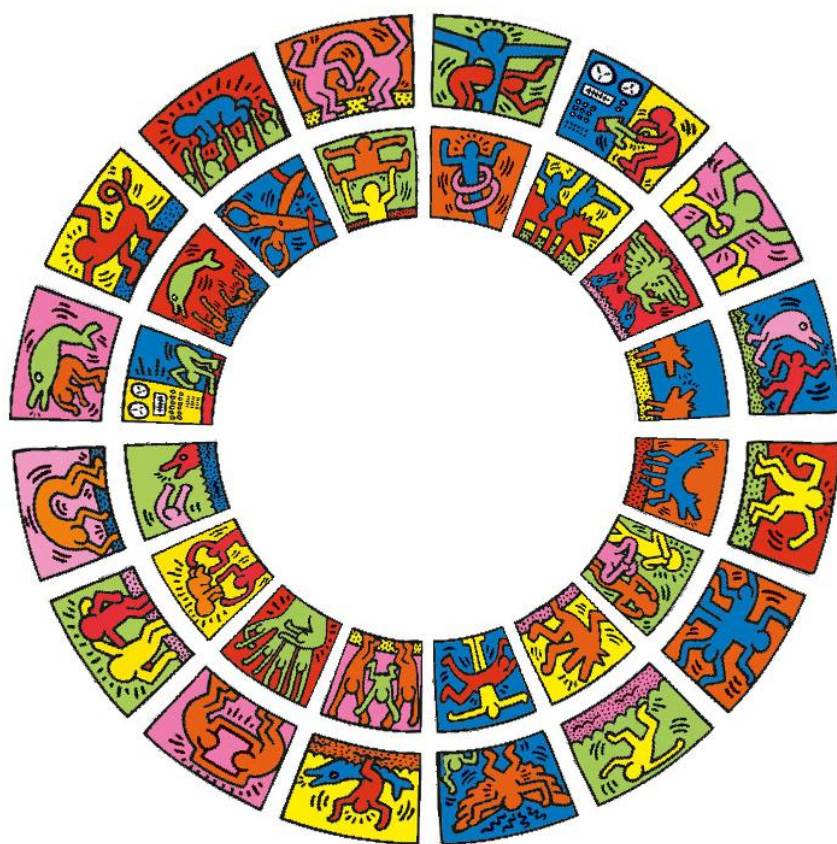


Figure 4: Keith Haring's Doodle.