How to draw ideogram

Zuguang Gu <z.gu@dkfz.de>

February 22, 2015

There is a new and more comprehensive vignette in this package which is focusing on genomic graphics. Nevertheless, this vignette is still useful for users to get a clue on how to draw genomic graphics by very basic low-level functions.

The most widely use of the circular layout is to display genomic information. In most circumstances, figure contains an ideogram. Drawing ideogram by **circlize** package is simple.

An ideogram is, in fact, a series of rectangles with different colors, so theoretically, you can use `circos.rect` to implement it. In the following example we draw the ideogram for human. The cytoband data for human can be downloaded from `http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/cytoBand.txt.gz` or from UCSC Table Browser (`http://genome-euro.ucsc.edu/cgi-bin/hgTables`). Uncompress the file and read it into R. Here **circlize** package already contains such file.

```
library(circlize)
d = read.table(file = paste0(system.file(package = "circlize"), "/extdata/cytoBand.txt"),
    colClasses = c("character", "numeric", "numeric", "character", "character"))
head(d)
```

```
##     V1       V2       V3     V4     V5
## 1 chr1        0  2300000 p36.33   gneg
## 2 chr1  2300000  5400000 p36.32 gpos25
## 3 chr1  5400000  7200000 p36.31   gneg
## 4 chr1  7200000  9200000 p36.23 gpos25
## 5 chr1  9200000 12700000 p36.22   gneg
## 6 chr1 12700000 16200000 p36.21 gpos50
```

In the data frame, the second column and the third column correspond to the intervals for cytogenetic bands.

Here, setting `colClasses` argument when reading the cytoband file is very important. Because positions on chromosomes are represented as large integers (the second column and third column), by default, `read.table` would store such data as `integer` mode. When arranging the layout, **circlize** will sum up such positions to determine the range of chromosomes and summation of such large integers would throw error of data overflow, thus you must set the data mode to floating point (`numeric`).

Since chromosomes are sorted by their names which are as mode of `character`, the default order would look like "chr1, chr10, chr11, ..., chr2, chr20, ...". We need to sort chromosomes by the numeric index first.

The process is simple. Extract the number part (1, 2, ..., 22) and the letter part (`X`, `Y`) in chromosome names. Sorted them separately and finally combine them back.

```
chromosome = unique(d[[1]])
chromosome.ind = gsub("chr", "", chromosome)
chromosome.num = grep("^\\d+$", chromosome.ind, value = TRUE)
chromosome.letter = chromosome.ind[!grepl("^\\d+$", chromosome.ind)]
chromosome.num = sort(as.numeric(chromosome.num))
chromosome.letter = sort(chromosome.letter)
chromosome.num = paste("chr", chromosome.num, sep = "")
chromosome.letter = paste("chr", chromosome.letter, sep = "")
```

```
chromosome = c(chromosome.num, chromosome.letter)
chromosome
```

```
##  [1] "chr1"  "chr2"  "chr3"  "chr4"  "chr5"  "chr6"  "chr7"  "chr8"
##  [9] "chr9"  "chr10" "chr11" "chr12" "chr13" "chr14" "chr15" "chr16"
## [17] "chr17" "chr18" "chr19" "chr20" "chr21" "chr22" "chrX"  "chrY"
```

The cytoband data also provides the range of each chromosome. This can be used to set as `xlim` of each chromosome. In the following code, we calculate the start position and the end position of each chromosome and store them in a matrix in which order of rows of `xlim` correspond to the order of elements in `chromosome`.

```
xlim = matrix(nrow = 0, ncol = 2)
for(chr in chromosome) {
    d2 = d[d[[1]] == chr, ]
    xlim = rbind(xlim, c(min(d2[[2]]), max(d2[[3]])))
}
```

Note that chromosome name in UCSC has prefix of 'chr', so if you are using chromosomes form 1000 Genome project which have no 'chr' prefix, remember to add it.

Before initializing the circular layout, we need to set some graphic parameters. Here we do not need any cell paddings and we do not need lines to be too thick because genomic graphics are always huge.

```
par(lwd = 0.5)
circos.par(cell.padding = c(0, 0, 0, 0))
```

In the initialization step, width of each sector corresponds to the length of each chromosome. Also the order of sectors is determined in this step. Here we must explicitly set the levels of the factors to make sure the order of chromosomes is "chr1, chr2, chr3, ..." or else the order would be the alphabetical which is "chr1, chr11, ...". After the initialization step, the position of each chromosome as well as the order are stored in an internal variable. So in the later step, as long as the chromosome is specified, graphics will be put in the right sector.

```
circos.initialize(factors = factor(chromosome, levels = chromosome), xlim = xlim)
```

After each chromosome has been allocated in the circle, we can draw the ideogram. Besides that, we also want to draw additional information such as axes and names of chromosomes. Here we will draw ideogram, axis and the chromosome names in one same track (It is just an option, also you can draw ideogram, axes and names of chromosomes in different tracks as you like). in the following code, we create the first track in which there are 24 cells and each cell corresponds to a chromosome. The x-range of each cell is the range of the chromosome and the y-range of each cell is from 0 to 1.

```
circos.trackPlotRegion(ylim = c(0, 1), bg.border = NA, track.height = 0.1)
```

In the above codes, without specifying `factors` argument, `circos.trackPlotRegion` will automatically create plotting regions for all available sectors which have already been initialized.

Now in each cell, we draw the ideogram for each chromosome. Code is simple. The steps are: for each chromosome:

1. assign different colors for different cytogenetic bands;
2. draw rectangle for different bands;
3. add axes;
4. add chromosome names.

Here the color theme is from `http://circos.ca/tutorials/course/slides/session-2.pdf`, page 42.

```
for(chr in chromosome) {
    # data in current `chr`
    d2 = d[d[[1]] == chr, ]
    n = nrow(d2)

    # assign colors
    col = rep("#FFFFFF", n)
    col[d2[[5]] == "gpos100"] = rgb(0, 0, 0, maxColorValue = 255)
    col[d2[[5]] == "gpos"]    = rgb(0, 0, 0, maxColorValue = 255)
    col[d2[[5]] == "gpos75"]  = rgb(130, 130, 130, maxColorValue = 255)
    col[d2[[5]] == "gpos66"]  = rgb(160, 160, 160, maxColorValue = 255)
    col[d2[[5]] == "gpos50"]  = rgb(200, 200, 200, maxColorValue = 255)
    col[d2[[5]] == "gpos33"]  = rgb(210, 210, 210, maxColorValue = 255)
    col[d2[[5]] == "gpos25"]  = rgb(200, 200, 200, maxColorValue = 255)
    col[d2[[5]] == "gvar"]    = rgb(220, 220, 220, maxColorValue = 255)
    col[d2[[5]] == "gneg"]    = rgb(255, 255, 255, maxColorValue = 255)
    col[d2[[5]] == "acen"]    = rgb(217, 47, 39, maxColorValue = 255)
    col[d2[[5]] == "stalk"]   = rgb(100, 127, 164, maxColorValue = 255)

    # rectangles for different locus
    for(i in seq_len(n)) {
        circos.rect(d2[i, 2], 0, d2[i, 3], 0.4, sector.index = chr,
            col = col[i], border = NA)
    }
    # rectangle that cover the whole chromosome
    circos.rect(d2[1, 2], 0, d2[n, 3], 0.4, sector.index = chr, border = "black")

    # axis
    major.at = seq(0, 10^nchar(max(xlim[, 2])), by = 50000000)
    circos.axis(h = 0.5, major.at = major.at,
        labels = paste(major.at/1000000, "MB", sep = ""),
        labels.facing = "clockwise", labels.niceFacing = TRUE,
        sector.index = chr, labels.cex = 0.2)
    chr.xlim = get.cell.meta.data("xlim", sector.index = chr)

    # chromosome names, only the number part or the letter part
    circos.text(mean(chr.xlim), 1.2, labels = gsub("chr", "", chr),
        sector.index = chr, cex = 0.8)
}
```

In the above code, you can find the `ylim` for the cells in the first track is `c(0, 1)` and the y-value in `circos.text` is `1.2` which exceeds the `ylim`. There may be some warnings saying some points are out of the plotting region. But in fact it is OK to draw something outside the plotting regions. You just need to make sure the final figure looks good.

If you do not want to draw ideogram in the most outside of the circos layout. You can draw it in other tracks as you wish.

As we introduced in the main vignettes, code in the `for` loop can also be put inside `panel.fun` when calling `circos.trackPlotRegion`.

If there is a translocation from position 111111111 in chromosome 2 to position 55555555 in chromosome 16. It can represent as a link in the circular layout.

```
circos.link("chr2", 111111111, "chr16", 55555555)
```

If position 88888888 in chromosome 6 is important and we want to mark it, we can first create a new track and add line and text in the specified cell. Note this track will overlap with the link which we added before, but since `bg.border` is set to `NA`, nothing will be plotted for this invisible track.

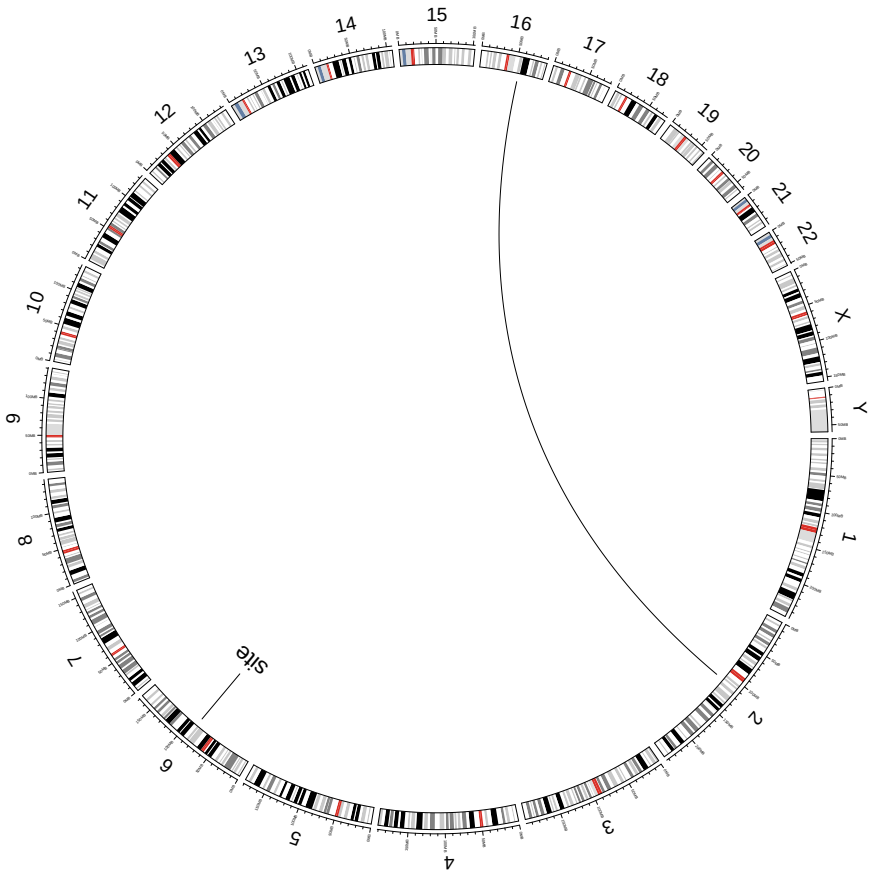

Figure 1: Ideogram in circular layout.

```
# create a new track
circos.trackPlotRegion(ylim = c(0, 1), bg.border = NA)
circos.text(88888888, 0.2, labels = "site", sector.index = "chr6", adj = c(0.5, 1))
circos.lines(c(88888888, 88888888), c(0.3, 1), sector.index = "chr6", straight = TRUE)
```

Finally, don't forget to call `circos.clear` in the end.

```
circos.clear()
```

For other tracks of genomic graphics, the genomic coordinate (positions on chromosomes) are x-values and measurements on genomic positions are y-values.

The final figure is figure 1.