How to draw ideogram

Zuguang Gu <z.gu@dkfz.de>

September 1, 2013

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

An ideogram is, in fact, a series of rectangles with different colors. In the following example we are going to draw the ideogram for human.

The cytoband data for human can be download from `http://hgdownload.cse.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 = paste(system.file(package = "circlize"),
+         "/extdata/cytoBand.txt", sep=""),
+     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 are intervals for cytogenetic bands.

Here, setting the `colClasses` argument when reading the cytoband file is very important, because the positions on chromosomes are large integers (the second column and third column), by default `read.table` would store such data as `integer` mode. The sumation of such large integers would throw error of data overflow. So you must set the data mode to floating point (`numeric`).

Since chromosomes are sorted by their names which are as mode of `character`, the 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`) of chromosome names. Sorted them seperately and finally combine them.

```
> 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"  "chr9"
[10] "chr10" "chr11" "chr12" "chr13" "chr14" "chr15" "chr16" "chr17" "chr18"
[19] "chr19" "chr20" "chr21" "chr22" "chrX"  "chrY"
```

The cytoband data also provides the range of each chromosome. This can be set as the `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`.

By the way, if you don't want to draw ideogram, reading the cytoband file is also useful because it tells you ranges of chromosomes and help you to allocate chromosomes in the circle.

```
> 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]])))
+ }
```

Since reading cytoband file is a general task, we embed the above code into a simple function `read.cytoband`. The function would return a list which includes a data frame with all cytoband data, sorted chromosome names and length of chromosomes.

```
> cytoband = read.cytoband()  # by default, it reads human cytoband data
> cytoband = read.cytoband(species = "hg19")  # or download from UCSC
> d = cytoband$df
> chromosome = cytoband$chromosome
> xlim = cbind(rep(0, length(chromosome)), cytoband$chr.len)
```

Before we draw the circular layout, we need to set some graphic parameters. Here we do not need any cell paddings and we do not need the line to be too thick because genomic graphic is always huge.

```
> par(mar = c(1, 1, 1, 1), lwd = 0.5)
> circos.par("cell.padding" = c(0, 0, 0, 0))
```

Initialize the circular layout with ranges of chromosomes. In the initialization step, width of each sector would correspond to the length of each chromosome. Also the order of sectors would be 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 would be put in the right sector which corresponds to the selected chromosome.

In the initialization step, order of the `xim` matrix should correspond to the order of `levels` of the `factors`, so do not be confused here.

```
> 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 the axis for chromosomes and names of chromosomes. Here we would draw ideogram, axis and the chromosome names in one track (It is just an option, also you can draw ideogram, axis 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(factors = chromosome, ylim = c(0, 1), bg.border = NA,
+     track.height = 0.1)
```

In the above codes, it is not necessary to set the `factors` argument. If `factors` is not set, `circos.trackPlotRegion` will automatically create plotting regions for all available sectors which have already been initialized. But explicitly specifying the `factors` argument would make your code more clear for reading. And the value for `factors` does not need to be a real factor. If it is not a factor, it would be converted to a factor internally. If the value for `factors` is already a factor, the level of the factor also does not need to be specified because the cells are positioned with the order of chromosomes which is already defined in the initialization step. (But to be rigorous, order of `factors` matters if graphics arguments like `bg.border` or `bg.col` is a vector with length larger than 1. However, in most circumstance, we do not need to set `bg.border` or `bg.col` as a vector, so you don't need to worry about it and just ignore this note).

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 `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 = ""),
+         sector.index = chr, labels.cex = 0.3)
+     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.

Also, the step of assigning colors to cytogenetic bands can be embed for repetitive use. Here a function called `cytoband.col` is provided in the package which can do such things.

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 circos layout.

```
> circos.link(sector.index1 = "chr2", point1 = 111111111, sector.index2 = "chr16",
+     point2 = 55555555)
```

If position 88888888 in chromosome 6 is important and we want to mark it, we can use following codes. First create a new track. Here there is no specifying of `factors`, thus the new track would create plotting regions for all available sectors (but with no borders. You would not see these cells but they really exist.). Note you can not create plotting region for a single cell, however you can write so, but in fact plotting region for cells in all sectors would be created.

```
> # 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)
```

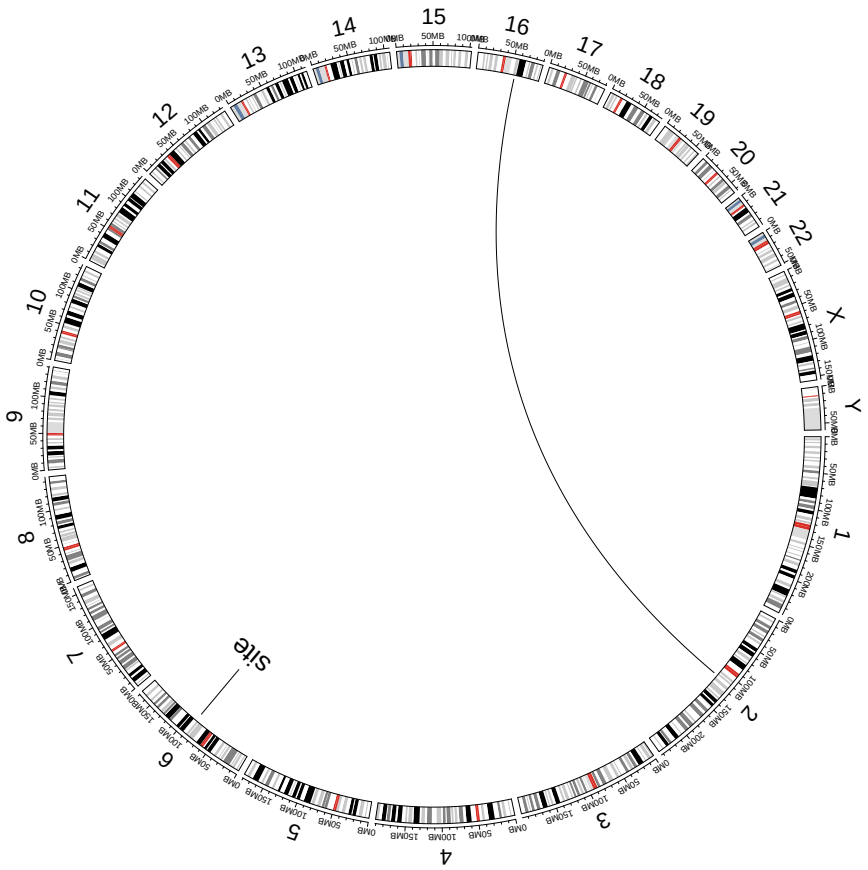

Figure 1: Ideogram in circular layout

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

The finnal figure looks like figure 1.

In the *circlize* package, there is already a `circos.initializeWithIdeogram` function to initialize circular layout with an ideogram. However, how to embed the ideogram into the circular layout is really subjective, such as the position and colors of the ideogram, or maybe only subset of chromosomes are going to be plotted, or maybe there are some zoomings for certain chromosomes (see `http://circos.ca/intro/features/`. 'GLOBAL AND LOCAL ZOOMING' section). So the `circos.initializeWithIdeogram` is not a full functional function, it is only an example function to show how to allocate sectors for chromosomes and how to draw ideogram. Thus users can draw their style of

ideogram according the above example codes. With such low level circos graphic functions provided in *circlize*, there would be no difficulty for users to implement their own genomic graphics. All you need to remember is that **complicated graphics are assembled by simple graphics**.

Finally, more informative and specialized genomic graphics are figure 2, figure 3 and figure 4. Figure 3 in fact combines two independent circos plots, and figure 4 re-defined sector width to put two zoomed chromosomes in half part of the circle. Users can refer to the main vignette to find out how to implement.

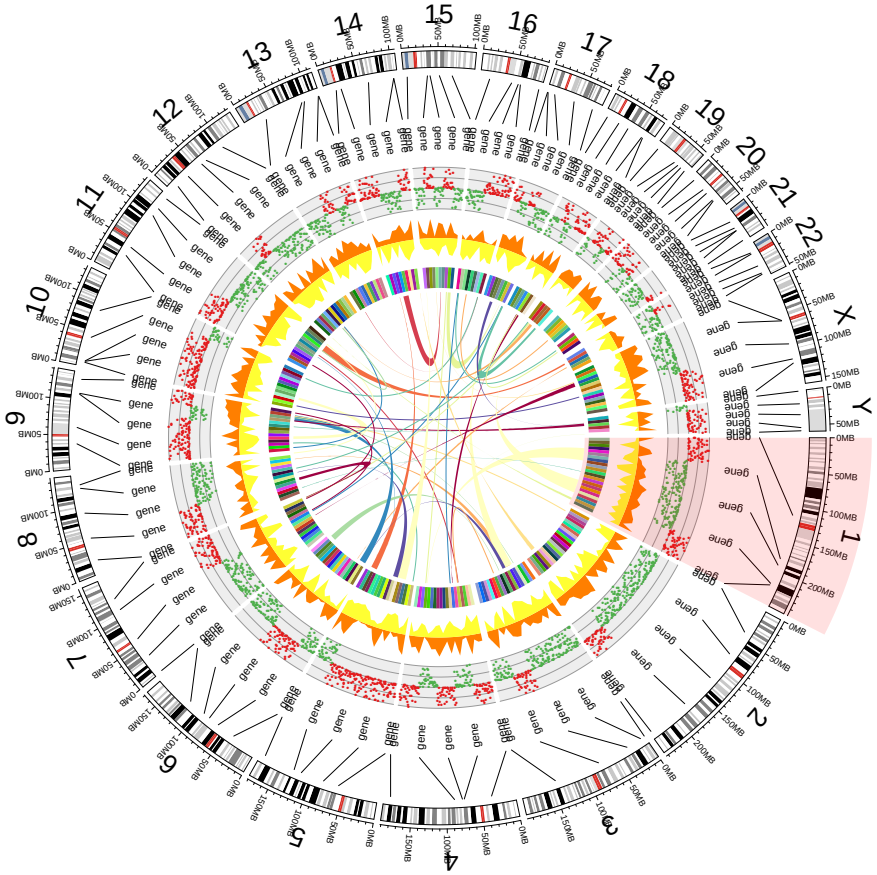

Figure 2: Detailed genomic graphic

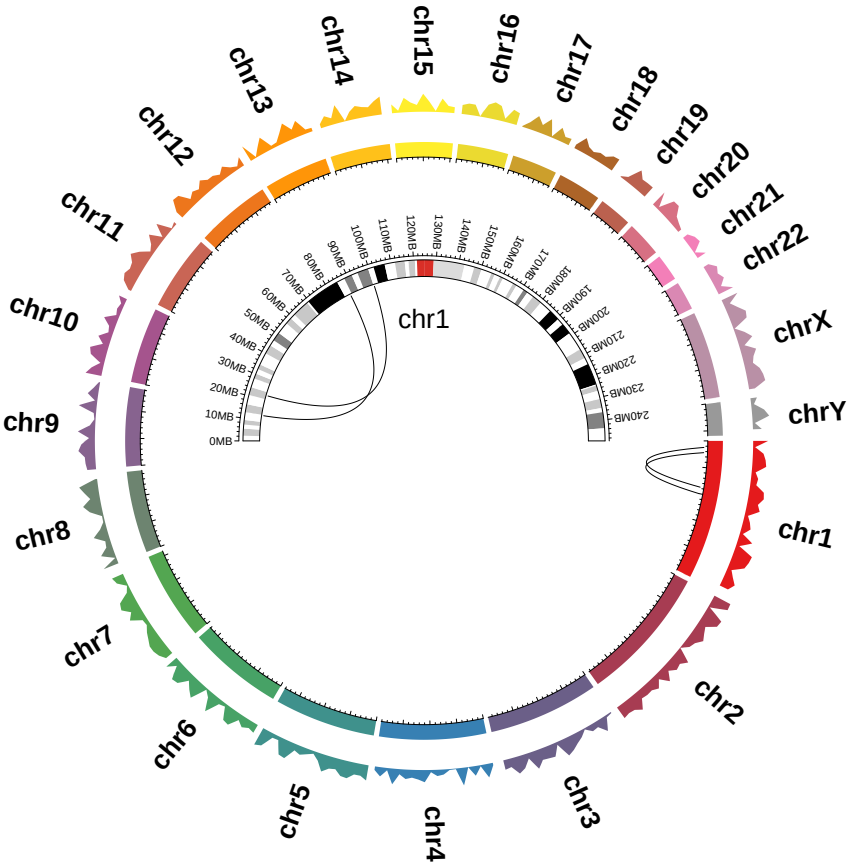

Figure 3: Two tracks of chromosomes in which the inner one zooms in chromosome 1 from the outer one.

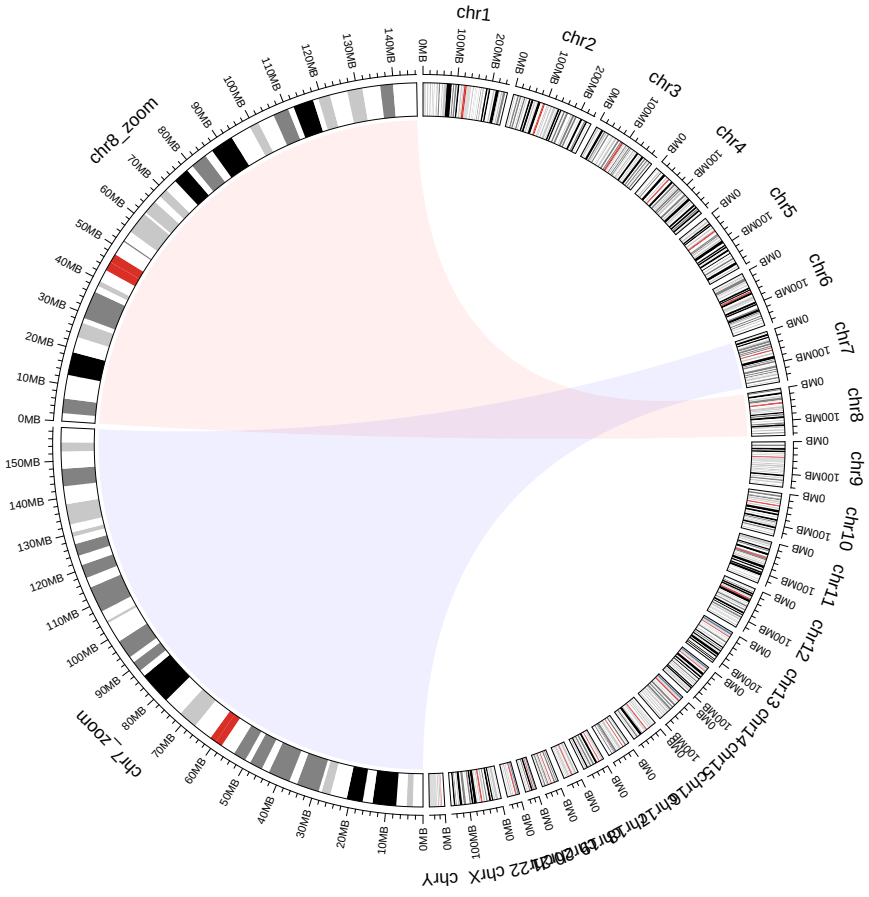

Figure 4: Chromosome 7 and 8 are zoomed in at the half of the circle