The OmicCircos usages by examples

Ying Hu and Chunhua Yan

October 24, 2023

Contents

1 Introduction

The OmicCircos package generates high-quality circular plots for visualizing variations in omics data. The data can be gene or chromosome position-based values from mutation, copy number, expression, and methylation analyses. This package is capable of displaying variations in scatterplots, lines, and text labels. The relationships between genomic features can be presented in the forms of polygons and curves. By utilizing the statistical and graphic functions in R/Bioconductor environment, OmicCircos is also able to draw boxplots, histograms, and heatmaps from multiple sample data. Each track is drawn independently, which allows the use to optimize the track quickly and easily.

In this vignette, we will introduce the package plotting functions using simulation data and TCGA gene expression and copy number variation (cnv) data (http://www.cancergenome.nih.gov/).

A quick way to load the vignette examples is:

```
vignette("OmicCircos")
```

2 Input file formats

Four input data files are used in the package: segment data, mapping data, link data and link polygon data. Segment data are required to draw the anchor circular track. The remaining three data sets are used to draw additional tracks or connections.

2.1 segment data

The `segment data` lay out the foundation of a circular graph and typically are used to draw the outmost anchor track. In the segment data, column 1 should be the segment or chromosome names. Columns 2 and 3 are the start and end positions of the segment. Columns 4 and 5 are optional which can contain additional description of the segment. The package comes with the segment data for human (hg18 and hg19) and mouse (mm9 and mm10). Let's start by loading the package

```
options(stringsAsFactors = FALSE);
library(OmicCircos);
## input hg19 cytogenetic band data
data(UCSC.hg19.chr);
head(UCSC.hg19.chr);
```

```
  chrom chromStart chromEnd   name gieStain
1  chr1          0  2300000 p36.33     gneg
2  chr1    2300000  5300000 p36.32   gpos25
3  chr1    5300000  7100000 p36.31     gneg
4  chr1    7100000  9200000 p36.23   gpos25
5  chr1    9200000 12600000 p36.22     gneg
6  chr1   12600000 16100000 p36.21   gpos50
```

2.2 mapping data

The `mapping data` are an R data frame which includes values to be drawn in the graph. In the mapping data, columns 1 and 2 are segment name and position respectively. Column 3 and beyond is optional which can be the value or name. In the following example, the third column is the gene symbol. Column 4 and 5 are the gene expression values for each sample.

```
options(stringsAsFactors = FALSE);
# load the OmicCircos-package
library(OmicCircos);
```

```
## TCGA gene expression data
data(TCGA.BC.gene.exp.2k.60);
head(TCGA.BC.gene.exp.2k.60[,c(1:5)]);
```

```
    chr        po     NAME TCGA.A1.A0SK.01A TCGA.A1.A0SO.01A
282  10 122272906 PPAPDC1A           -0.809            0.224
363  15  46973079     SHC4           -0.704            3.656
456  19  63014177   ZNF552           -3.116            0.417
15    1  67590402  IL12RB2            3.420            4.054
381  16   8750130     ABAT           -3.165           -1.880
238   8  87486702     WWP1           -1.713           -2.314
```

2.3 link data

The `link data` are for drawing curves between two anchor points. In the link data, columns 1, 2, 3 are the segment name, position, label of the first anchor point; columns 4, 5, 6 are segment name, position, label of the second anchor point Column 7 is optional and could be used for the link type description.

```
options(stringsAsFactors = FALSE);
# load the OmicCircos-package
library(OmicCircos);
## TCGA fusion gene data
data(TCGA.BC.fus);
head(TCGA.BC.fus[,c(1:6)]);
```

```
  chr1      po1    gene1 chr2       po2    gene2
1    2 63456333    WDPCP   10  37493749 ANKRD30A
2   18 14563374   PARD6G   21  14995400    POTED
3   10 37521495 ANKRD30A    3  49282645   CCDC36
4   10 37521495 ANKRD30A    7 100177212    LRCH4
5   18 18539803    ROCK1   18    112551   PARD6G
6   12  4618159  C12orf4   18   1514414   PARD6G
```

2.4 link polygon data

The `link polygon data` are for connecting two segments with a polygon graph. In the link polygon data, columns 1, 2 and 3 are the name, start and end points for the first segment and columns 4, 5 and 6 are the name, start and end points for the second segment.

3 The package functions

There are three main functions in the package: `sim.circos`, `segAnglePo` and `circos`. `sim.circos` generates simulation data for drawing circular plots. `segAnglePo` converts the genomic (linear) coordinates (chromosome base pair positions) to the angle based coordinates along circumference. circos enables users to superimpose graphics on the circle track.

3.1 sim.circos

The `sim.circos` function generates four simutatedinput data files, which allows users to preview the graph quickly with different parameters and design an optimal presentation with desired features. In the following example, there are 10 segments, 10 individuals, 10 links, and 10 link polygons. Each segment has the value ranging from 20 to 50. The values will be generated by rnorm(1) + i. The i is the ordinal number of the segments. The values are increased by the segment order.

```
options(stringsAsFactors = FALSE);
# load the OmicCircos-package
library(OmicCircos);
# set up the initial parameters
seg.num     <- 10;
ind.num     <- 20;
seg.po      <- c(20:50);
link.num    <- 10;
link.pg.num <- 10;
# run sim.circos function
sim.out     <- sim.circos(seg=seg.num, po=seg.po, ind=ind.num, link=link.num, link.pg=
    link.pg.num);
# display the data set names
names(sim.out)
# display the segment data
head(sim.out$seg.frame[,c(1:3)])
```

```
  seg.name seg.Start seg.End
1     chr1         0       1
2     chr1         1       2
3     chr1         2       3
4     chr1         3       4
5     chr1         4       5
6     chr1         5       6
```

```
# display the mapping data
head(sim.out$seg.mapping[,c(1:5)])
```

```
  seg.name seg.po  name1 name2  name3
1     chr1      1  0.045 0.852  0.299
2     chr1      2  0.052  1.15  1.681
3     chr1      3  1.394 0.346  0.377
4     chr1      4  1.678 0.268  2.028
5     chr1      5  2.099 1.366 -0.195
6     chr1      6 -0.339 0.734  2.343
```

```
# display the linking data
head(sim.out$seg.link)
```

```
  seg1 po1 name1  seg2 po2 name2 name3
1 chr9  37    n1  chr8  44    n1    n1
2 chr8  14    n2 chr10  31    n2    n2
3 chr5  39    n3  chr5  36    n3    n3
4 chr2  10    n4  chr5  37    n4    n4
5 chr4  12    n5  chr9   4    n5    n5
6 chr2   8    n6 chr10  23    n6    n6
```

```
# display the linking polygon data
head(sim.out$seg.link.pg)
```

```
  seg1 start1 end1 seg2 start2 end2
1 chr8     25   17 chr6     34    4
2 chr8     47   43 chr3      6   13
3 chr1     16    8 chr4     19   17
4 chr7      2   24 chr2     34   15
5 chr9      7    8 chr6      3   27
6 chr1     12    6 chr6     12    4
```

3.2 segAnglePo

The `segAnglePo` function converts the segment pointer positions (linear coordinates) into angle values (the angle based coordinates along circumference) and returns a data frame. It specifies the circle size, number of segments, and segment length.

```
library(OmicCircos);
options(stringsAsFactors = FALSE);
set.seed(1234);
## initial values for simulation data
seg.num     ← 10;
ind.num     ← 20;
seg.po      ← c(20:50);
link.num    ← 10;
link.pg.num ← 4;
## output simulation data
sim.out ← sim.circos(seg=seg.num, po=seg.po, ind=ind.num, link=link.num,
  link.pg=link.pg.num);
seg.f     ← sim.out$seg.frame;
seg.v     ← sim.out$seg.mapping;
link.v    ← sim.out$seg.link
link.pg.v ← sim.out$seg.link.pg
seg.num   ← length(unique(seg.f[,1]));
## select segments
seg.name ← paste("chr", 1:seg.num, sep="");
db       ← segAnglePo(seg.f, seg=seg.name);
```

```
      seg.name angle.start angle.end seg.sum.start seg.sum.end seg.start
 [1,] "chr1"   "270"       "315.398" "0"           "47"        "0"
 [2,] "chr2"   "317.398"   "361.83"  "47"          "93"        "0"
 [3,] "chr3"   "363.83"    "403.432" "93"          "134"       "0"
 [4,] "chr4"   "405.432"   "428.614" "134"         "158"       "0"
 [5,] "chr5"   "430.614"   "476.011" "158"         "205"       "0"
 [6,] "chr6"   "478.011"   "498.295" "205"         "226"       "0"
 [7,] "chr7"   "500.295"   "545.693" "226"         "273"       "0"
 [8,] "chr8"   "547.693"   "577.636" "273"         "304"       "0"
 [9,] "chr9"   "579.636"   "598.955" "304"         "324"       "0"
[10,] "chr10"  "600.955"   "628"     "324"         "352"       "0"
      seg.end
 [1,] "47"
 [2,] "46"
 [3,] "41"
 [4,] "24"
 [5,] "47"
 [6,] "21"
 [7,] "47"
 [8,] "31"
```

```
 [9,] "20"
[10,] "28"
```

In the above example, there are 10 segments in a circle. Column 1 is segment name. Columns 2, 3 are the start and end angles of the segment. Column 4 and 5 are the accumulative start and end positions. Column 6 and 7 are the start and end position for the segment. The plotting is clockwise starting at 12 o'clock (270 degree).

3.3 circos

The `circos` is the main function to draw different shapes of the circle. For example, expression and CNV data can be viewed using basic shapes like scatterplots and lines while structural variations such as translocations and fusion proteins can be viewed using curves and polygons to connect different segments. Additionally, multiple sample expression and CNV data sets can be displayed as boxplots, histograms, or heatmaps using standard R functions such as apply. The usage of this function is illustrated in the next section.

4 Plotting parameters

4.1 basic plotting

The input data sets were generated by textttsim.circos function.

```
options(stringsAsFactors = FALSE);
library(OmicCircos);
options(stringsAsFactors = FALSE);
set.seed(1234);

# initial
seg.num     ← 10;
ind.num     ← 20;
seg.po      ← c(20:50);
link.num    ← 10;
link.pg.num ← 4;

sim.out ← sim.circos(seg=seg.num, po=seg.po, ind=ind.num, link=link.num,
  link.pg=link.pg.num);

seg.f     ← sim.out$seg.frame;
seg.v     ← sim.out$seg.mapping;
link.v    ← sim.out$seg.link
link.pg.v ← sim.out$seg.link.pg
seg.num   ← length(unique(seg.f[,1]));

# name segment (option)
seg.name ← paste("chr", 1:seg.num, sep="");
db       ← segAnglePo(seg.f, seg=seg.name);
# set transparent colors
colors   ← rainbow(seg.num, alpha=0.5);
```

To get perfect circle, the output figure should be in square. The output file is the same width and height. The same line values are in the margin of the graphical parameters.

```
par(mar=c(2, 2, 2, 2));
plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="", main="");

circos(R=400, cir=db, type="chr",  col=colors, print.chr.lab=TRUE, W=4, scale=TRUE);
```

```
circos(R=360, cir=db, W=40, mapping=seg.v, col.v=3, type="l", B=TRUE, col=colors[1], lwd=2, scale=TRUE);
circos(R=320, cir=db, W=40, mapping=seg.v, col.v=3, type="ls", B=FALSE, col=colors[9], lwd=2, scale=TRUE);
circos(R=280, cir=db, W=40, mapping=seg.v, col.v=3, type="lh", B=TRUE, col=colors[7], lwd=2, scale=TRUE);
circos(R=240, cir=db, W=40, mapping=seg.v, col.v=19, type="ml", B=FALSE, col=colors, lwd=2, scale=TRUE);
circos(R=200, cir=db, W=40, mapping=seg.v, col.v=19, type="ml2", B=TRUE, col=colors, lwd=2);
circos(R=160, cir=db, W=40, mapping=seg.v, col.v=19, type="ml3", B=FALSE, cutoff=5, lwd=2);
circos(R=150, cir=db, W=40, mapping=link.v, type="link", lwd=2, col=colors[c(1,7)]);
circos(R=150, cir=db, W=40, mapping=link.pg.v, type="link.pg", lwd=2, col=sample(colors,link.pg.num));
```

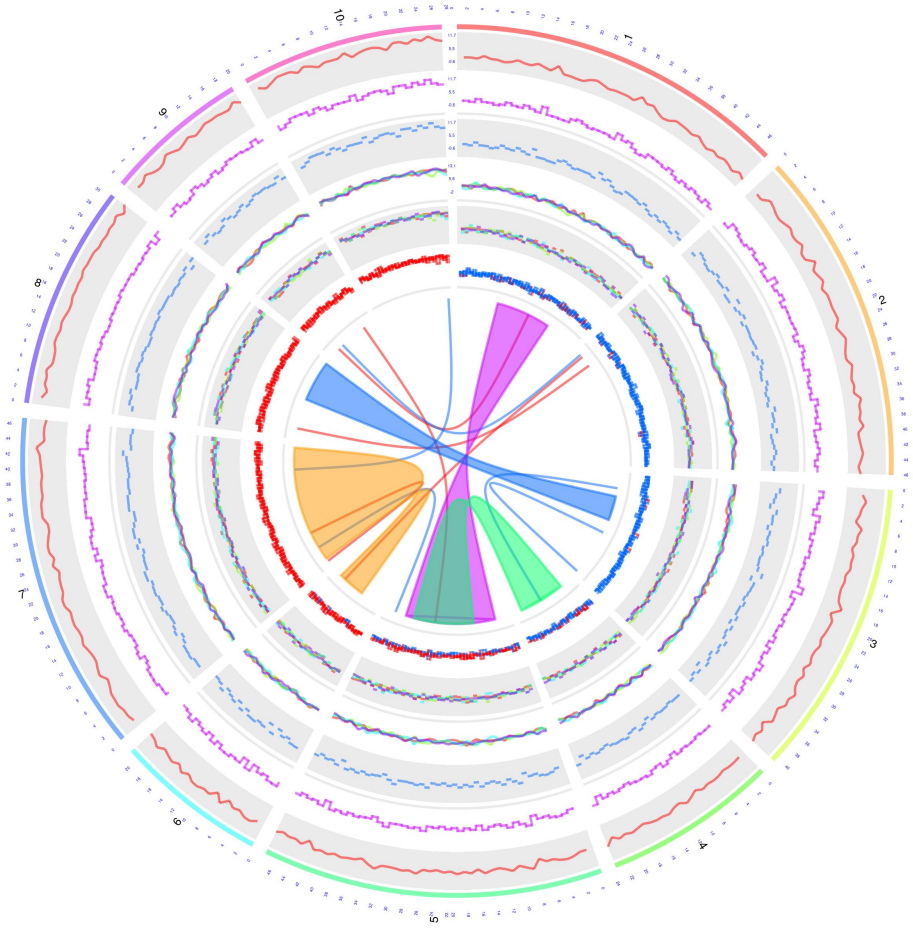

Figure 1

Figure 1 from outside to inside: Track is lines; Track 2 is the stair steps; Track 3 is the horizontal lines; Tracks 4, 5 and 6 are the multiple lines, stair steps and horizontal lines for multiple the samples.

```
options(stringsAsFactors = FALSE);
library(OmicCircos);
set.seed(1234);

## initial values for simulation data
seg.num      ← 10;
ind.num      ← 20;
seg.po       ← c(20:50);
link.num     ← 10;
link.pg.num ← 4;
## output simulation data
sim.out ← sim.circos(seg=seg.num, po=seg.po, ind=ind.num, link=link.num,
  link.pg=link.pg.num);

seg.f      ← sim.out$seg.frame;
seg.v      ← sim.out$seg.mapping;
link.v     ← sim.out$seg.link
link.pg.v ← sim.out$seg.link.pg
seg.num    ← length(unique(seg.f[,1]));

## select segments
seg.name ← paste("chr", 1:seg.num, sep="");
db       ← segAnglePo(seg.f, seg=seg.name);

colors   ← rainbow(seg.num, alpha=0.5);
```

```
par(mar=c(2, 2, 2, 2));
plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="", main="");

circos(R=400, type="chr", cir=db, col=colors, print.chr.lab=TRUE, W=4, scale=TRUE);
circos(R=360, cir=db, W=40, mapping=seg.v, col.v=8, type="box",  B=TRUE, col=colors
    [1], lwd=0.1, scale=TRUE);
circos(R=320, cir=db, W=40, mapping=seg.v, col.v=8, type="hist", B=TRUE, col=colors
    [3], lwd=0.1, scale=TRUE);
circos(R=280, cir=db, W=40, mapping=seg.v, col.v=8, type="ms",  B=TRUE, col=colors[7],
     lwd=0.1, scale=TRUE);
circos(R=240, cir=db, W=40, mapping=seg.v, col.v=3, type="h",  B=FALSE,  col=colors
    [2], lwd=0.1);
circos(R=200, cir=db, W=40, mapping=seg.v, col.v=3, type="s", B=TRUE, col=colors, lwd
    =0.1);
circos(R=160, cir=db, W=40, mapping=seg.v, col.v=3, type="b", B=FALSE, col=colors, lwd
    =0.1);
circos(R=150, cir=db, W=40, mapping=link.v, type="link", lwd=2, col=colors[c(1,7)]);
circos(R=150, cir=db, W=40, mapping=link.pg.v, type="link.pg", lwd=2, col=sample(
    colors,link.pg.num));
```

Figure 2 from outside to inside: Track 1 is the boxplot for the samples from column 8 (col.v=8) to the last column in the data frame seg.v with the scale; Track 2 and track 3 are the histograms (in horizontal) and the scatter plots for multiple samples as track 1. Tracks 4, 5 and 6 are the histogram (in vertical), scatter plot and vertical line for just one sample (column 3 in the data frame seg.v).

```
options(stringsAsFactors = FALSE);
library(OmicCircos);
set.seed(1234);

## initial values for simulation data
```

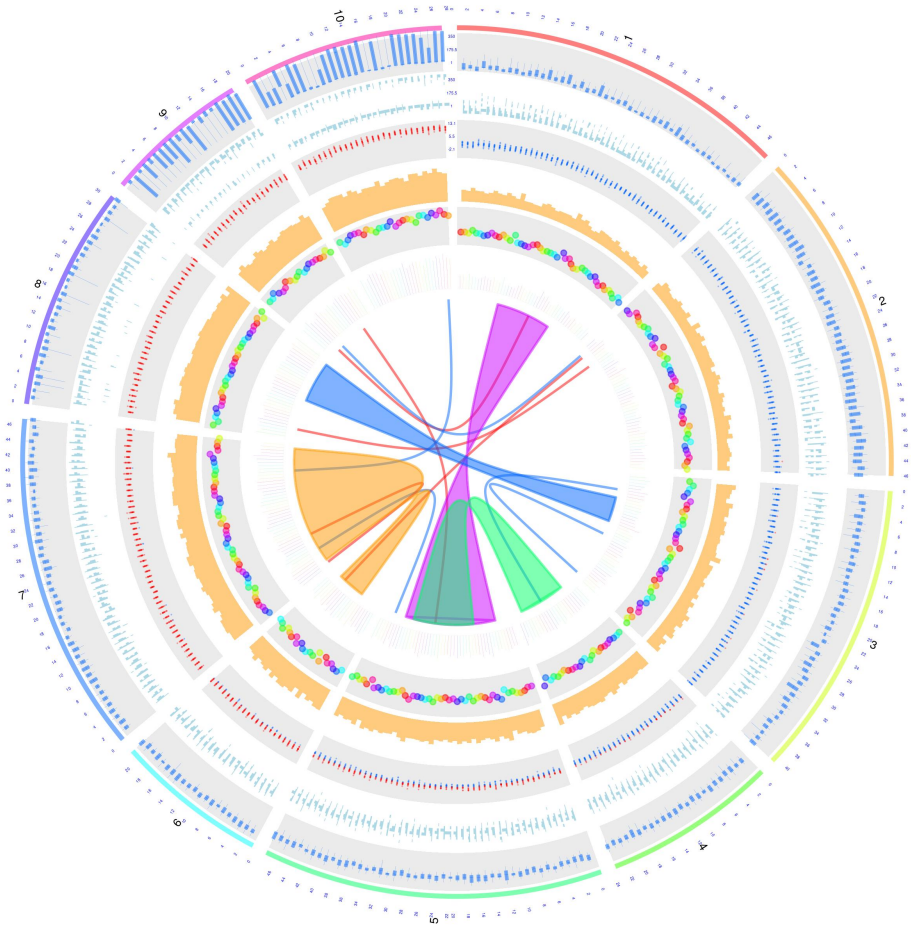

Figure 2

```
seg.num      ← 10;
ind.num      ← 20;
seg.po       ← c(20:50);
link.num     ← 10;
link.pg.num ← 4;
## output simulation data
sim.out ← sim.circos(seg=seg.num, po=seg.po, ind=ind.num, link=link.num,
  link.pg=link.pg.num);

seg.f      ← sim.out$seg.frame;
seg.v      ← sim.out$seg.mapping;
link.v     ← sim.out$seg.link
link.pg.v ← sim.out$seg.link.pg
seg.num    ← length(unique(seg.f[,1]));

##
seg.name ← paste("chr", 1:seg.num, sep="");
db         ← segAnglePo(seg.f, seg=seg.name);

colors    ← rainbow(seg.num, alpha=0.5);
```

```
par(mar=c(2, 2, 2, 2));
plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="", main="");

circos(R=400, type="chr", cir=db, col=colors, print.chr.lab=TRUE, W=4, scale=TRUE);
circos(R=360, cir=db, W=40, mapping=seg.v, col.v=8, type="quant90", B=FALSE, col=
    colors, lwd=2, scale=TRUE);
circos(R=320, cir=db, W=40, mapping=seg.v, col.v=3, type="sv", B=TRUE, col=colors[7],
     scale=TRUE);
circos(R=280, cir=db, W=40, mapping=seg.v, col.v=3, type="ss", B=FALSE, col=colors[3],
      scale=TRUE);
circos(R=240, cir=db, W=40, mapping=seg.v, col.v=8, type="heatmap", lwd=3);
circos(R=200, cir=db, W=40, mapping=seg.v, col.v=3, type="s.sd", B=FALSE, col=colors
    [4]);
circos(R=160, cir=db, W=40, mapping=seg.v, col.v=3, type="ci95", B=TRUE, col=colors
    [4], lwd=2);
circos(R=150, cir=db, W=40, mapping=link.v, type="link", lwd=2, col=colors[c(1,7)]);
circos(R=150, cir=db, W=40, mapping=link.pg.v, type="link.pg", lwd=2, col=sample(
    colors,link.pg.num));

the.col1=rainbow(10, alpha=0.5)[3];
highlight ← c(160, 410, 6, 2, 6, 10, the.col1, the.col1);
circos(R=110, cir=db, W=40, mapping=highlight, type="hl", lwd=1);

the.col1=rainbow(10, alpha=0.1)[3];
the.col2=rainbow(10, alpha=0.5)[1];
highlight ← c(160, 410, 3, 12, 3, 20, the.col1, the.col2);
circos(R=110, cir=db, W=40, mapping=highlight, type="hl", lwd=2);
```

Figure 3 from outside to inside: Track 1 is the three lines for quantile values for the samples from column 8 (col.v=8) to the last column in the data frame seg.v with the scale. The middle line is for the median, the outside line and the inside line are for 90% and the 10%, respectively; Track 2 is the circle points with the center=median and radium=variance; Track 3 is the circle plot with the center equal to the mean and scaled value (for example, the range from 0 to 3); Tracks 4 is the heatmap for the samples from column 8 (col.v=8) to the last column in the data frame seg.v; Track 5 is the circle plot with the center=median and radius=standard deviation; Track 6 is the 95% confidence

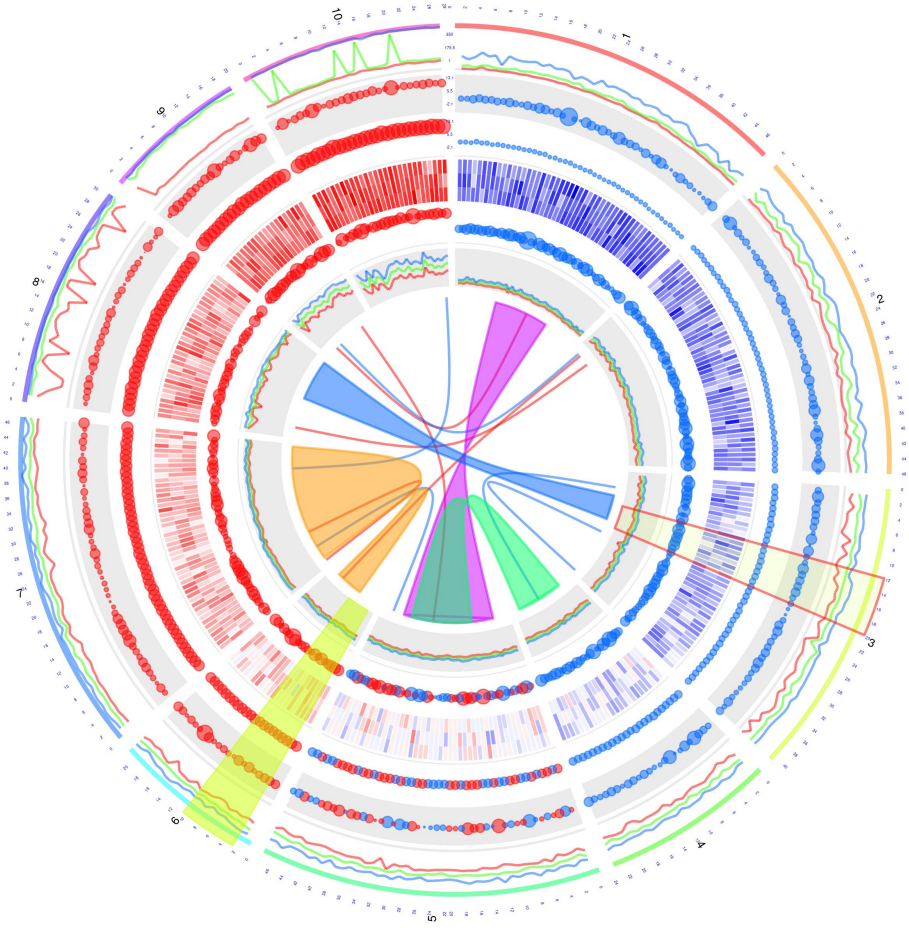

Figure 3

interval of the samples.

4.2 annotation

```
options(stringsAsFactors = FALSE);
library(OmicCircos);
set.seed(1234);

## load mm cytogenetic band data
data("UCSC.mm10.chr", package="OmicCircos");
ref     <- UCSC.mm10.chr;
ref[,1] <- gsub("chr", "", ref[,1]);
## initial values for simulation data
colors <- rainbow(10, alpha=0.8);
lab.n  <- 50;
cnv.n  <- 200;
arc.n  <- 30;
fus.n  <- 10;

## make arc data
arc.d <- c();
for (i in 1:arc.n){
  chr     <- sample(1:19, 1);
  chr.i   <- which(ref[,1]==chr);
  chr.arc <- ref[chr.i,];
  arc.i   <- sample(1:nrow(chr.arc), 2);
  arc.d   <- rbind(arc.d, c(chr.arc[arc.i[1],c(1,2)], chr.arc[arc.i[2],c(2,4)]));
}
colnames(arc.d) <- c("chr", "start", "end", "value");

## make fusion
fus.d <- c();
for (i in 1:fus.n){
  chr1    <- sample(1:19, 1);
  chr2    <- sample(1:19, 1);
  chr1.i  <- which(ref[,1]==chr1);
  chr2.i  <- which(ref[,1]==chr2);
  chr1.f  <- ref[chr1.i,];
  chr2.f  <- ref[chr2.i,];
  fus1.i  <- sample(1:nrow(chr1.f), 1);
  fus2.i  <- sample(1:nrow(chr2.f), 1);
  n1      <- paste0("geneA", i);
  n2      <- paste0("geneB", i);
  fus.d   <- rbind(fus.d, c(chr1.f[fus1.i,c(1,2)], n1, chr2.f[fus2.i,c(1,2)], n2));
}
colnames(fus.d) <- c("chr1", "po1", "gene1", "chr2", "po2", "gene2");

cnv.i <- sample(1:nrow(ref), cnv.n);
vale  <- rnorm(cnv.n);
cnv.d <- data.frame(ref[cnv.i,c(1,2)], value=vale);
```

```
par(mar=c(2, 2, 2, 2));
plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="");

```

```
circos(R=400, type="chr", cir="mm10", print.chr.lab=TRUE, W=4, scale=TRUE);
circos(R=340, cir="mm10", W=60, mapping=cnv.d, type="b3", B=TRUE, col=colors[7]);
circos(R=340, cir="mm10", W=60, mapping=cnv.d, type="s2", B=FALSE, col=colors[1], cex=0.5);
circos(R=280, cir="mm10", W=60, mapping=arc.d, type="arc2", B=FALSE, col=colors, lwd=10, cutoff=0);
circos(R=220, cir="mm10", W=60, mapping=cnv.d, col.v=3, type="b2", B=TRUE, cutoff=-0.2, col=colors[c(7,9)], lwd=2);
circos(R=160, cir="mm10", W=60, mapping=arc.d, col.v=4, type="arc", B=FALSE, col=colors[c(1,7)], lwd=4, scale=TRUE);
circos(R=150, cir="mm10", W=10, mapping=fus.d, type="link", lwd=2, col=colors[c(1,7,9)]);
```

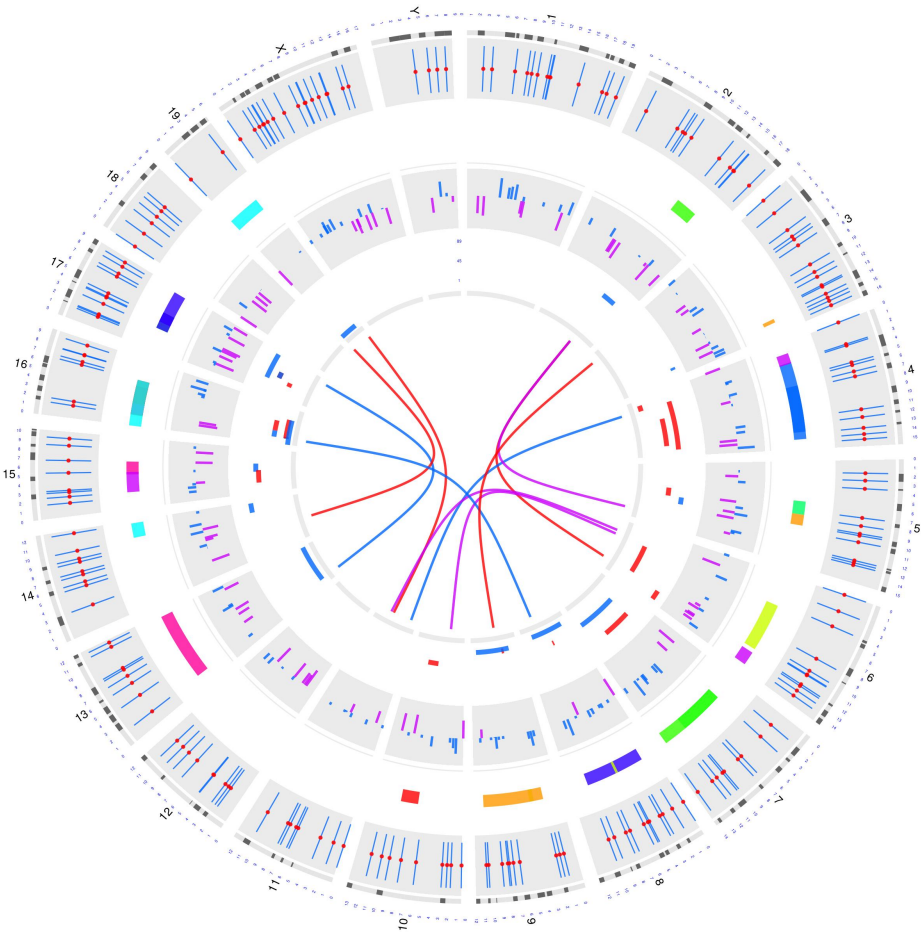

Figure 4

4.3 label

Figure 4 from outside to inside: Track 1 is the vertical lines with the same length and radius which can be used for the annotation of SNP positions; Track 2 is the arcs with the same radius which can be used for the segment annotation,

e.g. cnv (copy number variation); Track 3 is the barplot with positive and negative values; Track 4 is the arcs in the different radius.

```
options(stringsAsFactors = FALSE);
library(OmicCircos);

data("TCGA.PAM50_genefu_hg18");
data("TCGA.BC.fus");
data("TCGA.BC.cnv.2k.60");
data("TCGA.BC.gene.exp.2k.60");
data("TCGA.BC.sample60");
data("TCGA.BC_Her2_cnv_exp");

pvalue ← -1 * log10(TCGA.BC_Her2_cnv_exp[,5]);
pvalue ← cbind(TCGA.BC_Her2_cnv_exp[,c(1:3)], pvalue);

Her2.i ← which(TCGA.BC.sample60[,2] == "Her2");
Her2.n ← TCGA.BC.sample60[Her2.i,1];

Her2.j ← which(colnames(TCGA.BC.cnv.2k.60) %in% Her2.n);
cnv    ← TCGA.BC.cnv.2k.60[,c(1:3,Her2.j)];
cnv.m  ← cnv[,c(4:ncol(cnv))];
cnv.m[cnv.m >  2] ← 2;
cnv.m[cnv.m < -2] ← -2;
cnv ← cbind(cnv[,1:3], cnv.m);

Her2.j   ← which(colnames(TCGA.BC.gene.exp.2k.60) %in% Her2.n);
gene.exp ← TCGA.BC.gene.exp.2k.60[,c(1:3,Her2.j)];
colors ← rainbow(10, alpha=0.5);
```

```
par(mar=c(2, 2, 2, 2));
plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="");

circos(R=300, type="chr", cir="hg18", print.chr.lab=FALSE, W=4);
circos(R=310, cir="hg18", W=20, mapping=TCGA.PAM50_genefu_hg18, type="label",
       side="out", col=c("black", "blue","red"), cex=0.4);
circos(R=250, cir="hg18", W=50, mapping=cnv, col.v=4, type="ml3", B=FALSE, col=colors
    [7], cutoff=0, scale=TRUE);
circos(R=200, cir="hg18", W=50, mapping=gene.exp, col.v=4, type="ml3", B=TRUE, col=
    colors[3], cutoff=0, scale=TRUE);
circos(R=140, cir="hg18", W=50, mapping=pvalue, col.v=4, type="l", B=FALSE, col=colors
    [1], scale=TRUE);
## set fusion gene colors
cols  ← rep(colors[7], nrow(TCGA.BC.fus));
col.i ← which(TCGA.BC.fus[,1]==TCGA.BC.fus[,4]);
cols[col.i] ← colors[1];
circos(R=132, cir="hg18", W=50, mapping=TCGA.BC.fus, type="link", col=cols, lwd=2);
```

Figure 5 is an example of adding outside labels.

```
par(mar=c(2, 2, 2, 2));
plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="", main="");
circos(R=300, type="chr", cir="hg18", col=TRUE, print.chr.lab=FALSE, W=4);
circos(R=290, cir="hg18", W=20, mapping=TCGA.PAM50_genefu_hg18, type="label", side="in
    ", col=c("black", "blue"), cex=0.4);
circos(R=310, cir="hg18", W=50, mapping=cnv, col.v=4, type="ml3", B=TRUE, col=colors
    [7], cutoff=0, scale=TRUE);
```

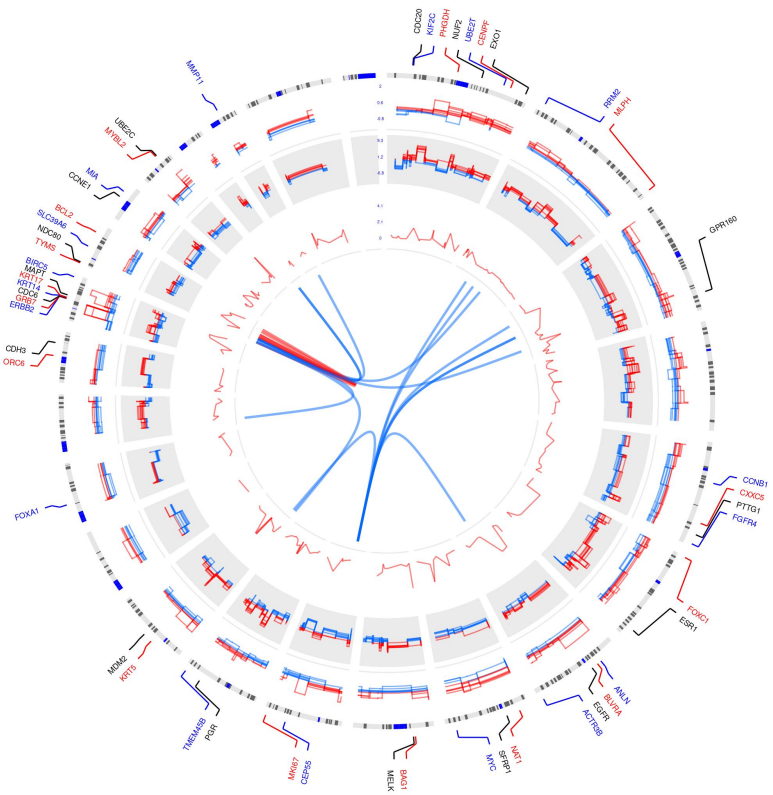
CDC20
KIF2C
PHGDH
NUF2
UBE2T
CENPF
EXO1
RRM2
MLPH
GPR160
CCNB1
CXXC5
PTTG1
FGFR4
FOXC1
ESR1
ANLN
BLVRA
EGFR
ACTR3B
MAPT
SFRP1
MYC
BUB1
MELK
MKI67
CEP55
TMEM45B
PGR
MDM2
KRT5
FOXA1
CDH3
ORC6
BIRC5
KRT17
CDC6
GSN
ERBB2
BCL2
SLC39A6
NDC80
TYMS
MIA
CCNE1
UBE2C
MYBL2
MMP11

Figure 5

```
circos(R=150, cir="hg18", W=50, mapping=gene.exp, col.v=4, type="ml3", B=TRUE, col=colors[3], cutoff=0, scale=TRUE);
circos(R=90, cir="hg18", W=50, mapping=pvalue, col.v=4, type="l", B=FALSE, col=colors[1], scale=TRUE);
circos(R=82, cir="hg18", W=50, mapping=TCGA.BC.fus, type="link", col=cols, lwd=2);
```

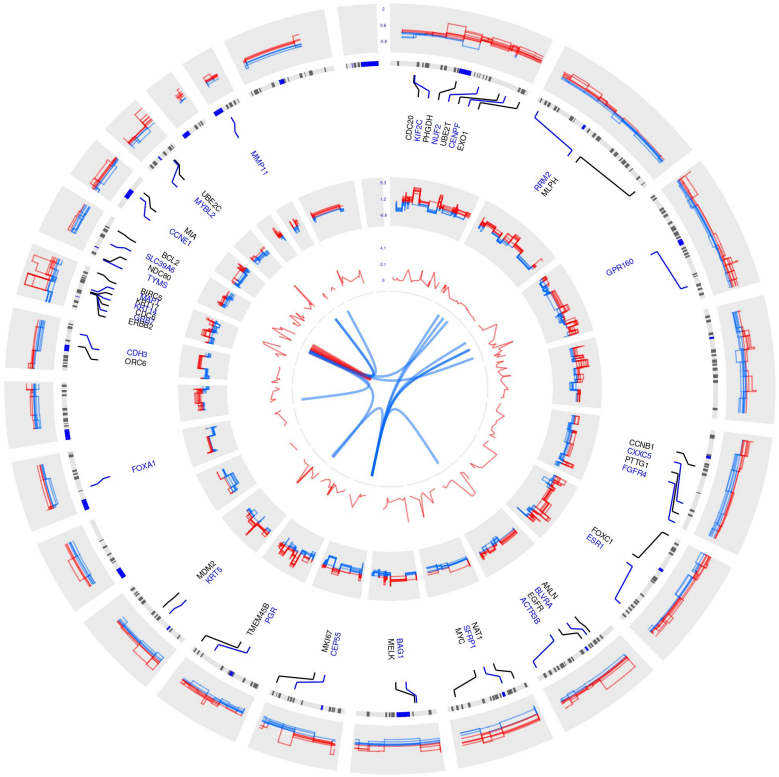

Figure 6

Figure 6 is an example of the inside labels.

4.4 heatmap

```
options(stringsAsFactors = FALSE);
library(OmicCircos);

data("TCGA.PAM50_genefu_hg18");
data("TCGA.BC.fus");
data("TCGA.BC.cnv.2k.60");
data("TCGA.BC.gene.exp.2k.60");
data("TCGA.BC.sample60");
```

```
data("TCGA.BC_Her2_cnv_exp");

pvalue ← -1 * log10(TCGA.BC_Her2_cnv_exp[,5]);
pvalue ← cbind(TCGA.BC_Her2_cnv_exp[,c(1:3)], pvalue);

Her2.i ← which(TCGA.BC.sample60[,2] == "Her2");
Her2.n ← TCGA.BC.sample60[Her2.i,1];

Her2.j ← which(colnames(TCGA.BC.cnv.2k.60) %in% Her2.n);
cnv    ← TCGA.BC.cnv.2k.60[,c(1:3,Her2.j)];
cnv.m  ← cnv[,c(4:ncol(cnv))];
cnv.m[cnv.m >  2] ← 2;
cnv.m[cnv.m < -2] ← -2;
cnv ← cbind(cnv[,1:3], cnv.m);

Her2.j   ← which(colnames(TCGA.BC.gene.exp.2k.60) %in% Her2.n);
gene.exp ← TCGA.BC.gene.exp.2k.60[,c(1:3,Her2.j)];

colors ← rainbow(10, alpha=0.5);
```

```
par(mar=c(2, 2, 2, 2));

plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="", main="");

circos(R=400, cir="hg18", W=4,   type="chr", print.chr.lab=TRUE, scale=TRUE);
circos(R=300, cir="hg18", W=100, mapping=gene.exp, col.v=4, type="heatmap2",
       cluster=TRUE, col.bar=TRUE, lwd=0.1, col="blue");
circos(R=220, cir="hg18", W=80,  mapping=cnv,  col.v=4,  type="ml3", B=FALSE, lwd=1,
     cutoff=0);
circos(R=140, cir="hg18", W=80,  mapping=pvalue, col.v=4,  type="l",  B=TRUE, lwd
    =1, col=colors[1]);

cols         ← rep(colors[7], nrow(TCGA.BC.fus));
col.i        ← which(TCGA.BC.fus[,1]==TCGA.BC.fus[,4]);
cols[col.i] ← colors[1];
circos(R=130, cir="hg18", W=10,  mapping=TCGA.BC.fus, type="link2", lwd=2, col=cols);
```

Figure 7: An example of a circular plots generated by OmicCircos showing the expression, CNV and fusion protein in 15 Her2 subtype samples from TCGA Breast Cancer data. Circular tracks from outside to inside: genome positions by chromosomes (black lines are cytobands); expression heatmap (red: up-regulated; blue: down-regulated); CNVs (red: gain; blue: loss); correlation p values between expression and CNVs; fusion genes.

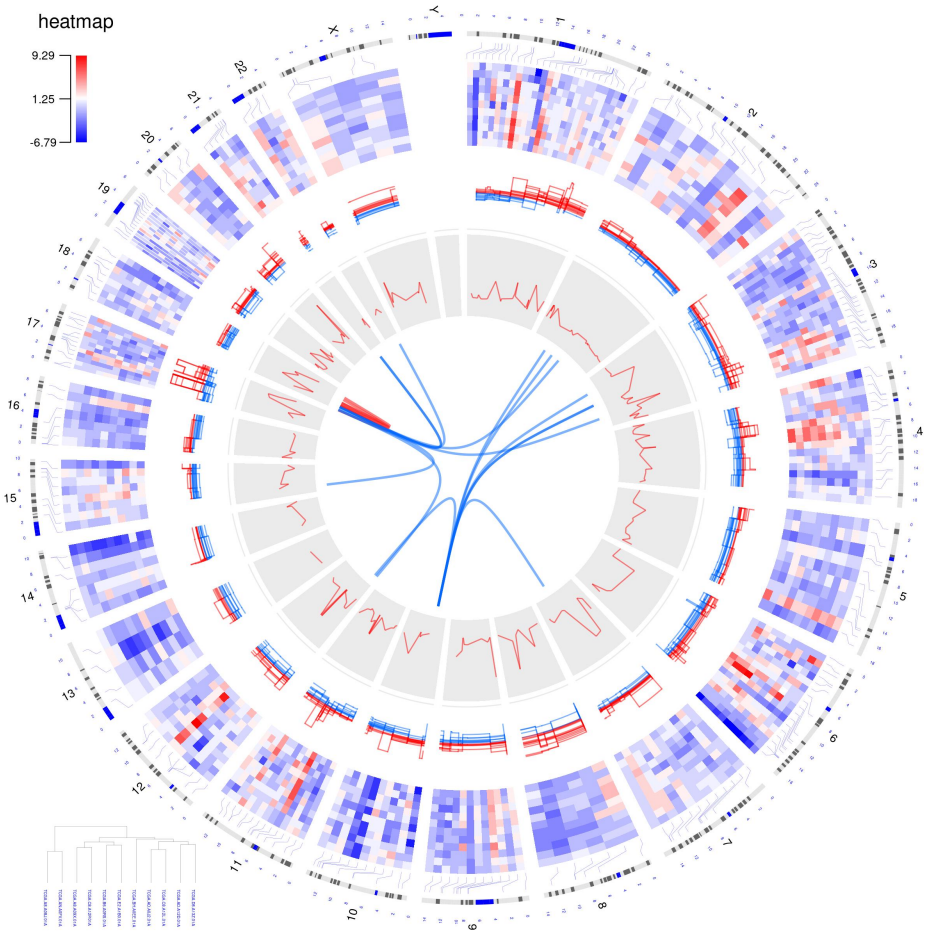

Figure 7