

# The OmicCircos usages by examples

Ying Hu and Chunhua Yan

October 29, 2024

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Input file formats</b>	<b>2</b>
2.1	segment data . . . . .	2
2.2	mapping data . . . . .	2
2.3	link data . . . . .	3
2.4	link polygon data . . . . .	3
<b>3</b>	<b>The package functions</b>	<b>3</b>
3.1	sim.circos . . . . .	3
3.2	segAnglePo . . . . .	5
3.3	circos . . . . .	6
<b>4</b>	<b>Plotting parameters</b>	<b>6</b>
4.1	basic plotting . . . . .	6
4.2	annotation . . . . .	10
4.3	label . . . . .	13
4.4	heatmap . . . . .	17

# 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:

```
1 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

```
1 options(stringsAsFactors = FALSE);
2 library(OmicCircos);
3 ## input hg19 cytogenetic band data
4 data(UCSC.hg19.chr);
5 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
```

### 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.

```
1 options(stringsAsFactors = FALSE);
2 # load the OmicCircos-package
3 library(OmicCircos);
4 ## TCGA gene expression data
```

```

5 data(TCGA.BC.gene.exp.2k.60);
6 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

## 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.

```

1 options(stringsAsFactors = FALSE);
2 # load the OmicCircos-package
3 library(OmicCircos);
4 ## TCGA fusion gene data
5 data(TCGA.BC.fus);
6 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

## 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 simulated input 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.

```

1 options(stringsAsFactors = FALSE);
2 # load the OmicCircos-package
3 library(OmicCircos);
4 # set up the initial parameters
5 seg.num ← 10;
6 ind.num ← 20;
7 seg.po ← c(20:50);
8 link.num ← 10;
9 link.pg.num ← 10;
10 # run sim.circos function
11 sim.out ← sim.circos(seg=seg.num, po=seg.po, ind=ind.num, link=link.num, link.pg=
    link.pg.num);
12 # display the data set names
13 names(sim.out)
14 # display the segment data
15 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

```

```

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

```

```

    seg.name seg.po name1 name2 name3
1   chr1      1 -0.239 2.774 2.307
2   chr1      2 -1.466 1.043 -0.268
3   chr1      3  2.164 1.207 0.336
4   chr1      4  0.652 1.483 0.583
5   chr1      5  2.565 0.049 -0.226

```

```

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

```

```

    seg1 po1 name1 seg2 po2 name2 name3
1 chr10 14   n1 chr3 21   n1   n1
2 chr2  6   n2 chr6  9   n2   n2
3 chr7  3   n3 chr2  1   n3   n3
4 chr4  2   n4 chr9 35   n4   n4
5 chr1 14   n5 chr9 40   n5   n5

```

```

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

```

```

    seg1 start1 end1 seg2 start2 end2
1 chr8    19   27 chr4    20   15
2 chr5    15   16 chr8     0   27
3 chr9    30   44 chr6     2   21
4 chr5    18    8 chr2    17    2
5 chr9    22    9 chr5    17    5

```

## 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.

```
1 library(OmicCircos);
2 options(stringsAsFactors = FALSE);
3 set.seed(1234);
4 ## initial values for simulation data
5 seg.num     <- 10;
6 ind.num     <- 20;
7 seg.po      <- c(20:50);
8 link.num    <- 10;
9 link.pg.num <- 4;
10 ## output simulation data
11 sim.out <- sim.circos(seg=seg.num, po=seg.po, ind=ind.num, link=link.num,
12   link.pg=link.pg.num);
13 seg.f     <- sim.out$seg.frame;
14 seg.v     <- sim.out$seg.mapping;
15 link.v    <- sim.out$seg.link
16 link.pg.v <- sim.out$seg.link.pg
17 seg.num   <- length(unique(seg.f[,1]));
18 ## select segments
19 seg.name  <- paste("chr", 1:seg.num, sep="");
20 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"
```

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 `texttttsim.circos` function.

```
1 options(stringsAsFactors = FALSE);
2 library(OmicCircos);
3 options(stringsAsFactors = FALSE);
4 set.seed(1234);
5
6 # initial
7 seg.num ← 10;
8 ind.num ← 20;
9 seg.po ← c(20:50);
10 link.num ← 10;
11 link.pg.num ← 4;
12
13 sim.out ← sim.circos(seg=seg.num, po=seg.po, ind=ind.num, link=link.num,
14 link.pg=link.pg.num);
15
16 seg.f ← sim.out$seg.frame;
17 seg.v ← sim.out$seg.mapping;
18 link.v ← sim.out$seg.link
19 link.pg.v ← sim.out$seg.link.pg
20 seg.num ← length(unique(seg.f[,1]));
21
22 # name segment (option)
23 seg.name ← paste("chr", 1:seg.num, sep="");
24 db ← segAnglePo(seg.f, seg=seg.name);
25 # set transparent colors
26 colors ← rainbow(seg.num, alpha=0.5);
```

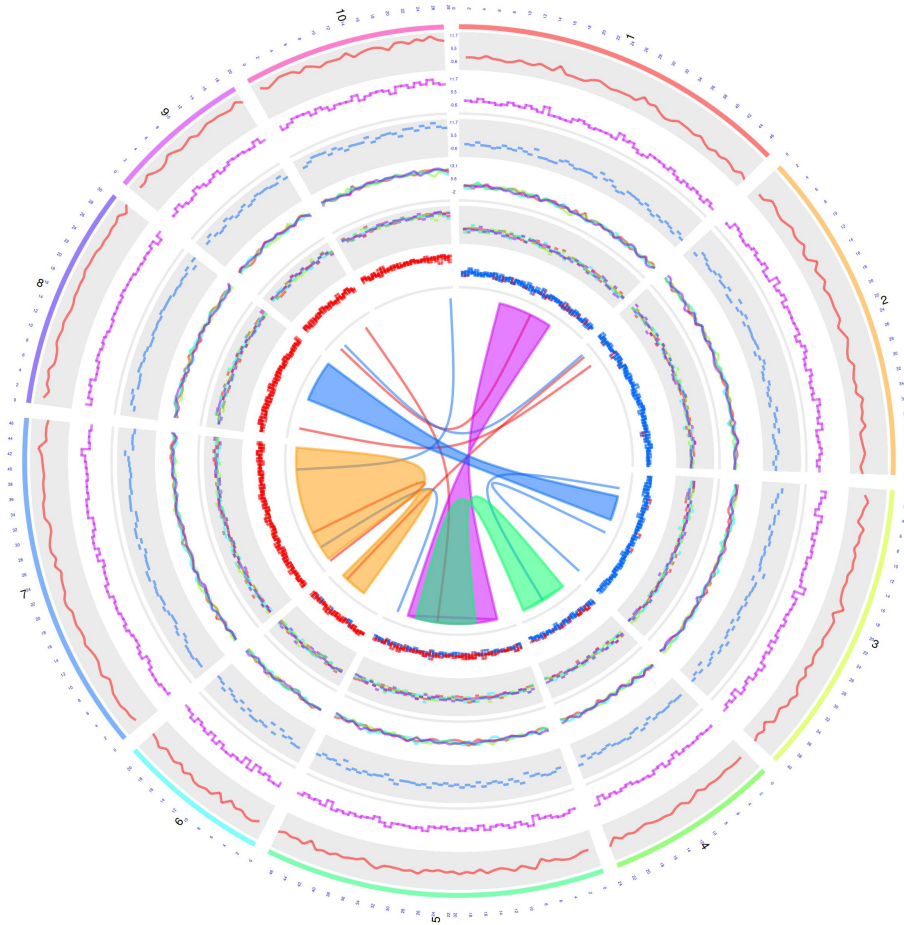
To get perfect circle, the output figure should be in square. The output file is the same width and height. The same line values are in the margin of the graphical parameters.

```
1 par(mar=c(2, 2, 2, 2));
2 plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="", main="");
3
4 circos(R=400, cir=db, type="chr", col=colors, print.chr.lab=TRUE, W=4, scale=TRUE);
5 circos(R=360, cir=db, W=40, mapping=seg.v, col.v=3, type="l", B=TRUE, col=colors[1],
6 lwd=2, scale=TRUE);
7 circos(R=320, cir=db, W=40, mapping=seg.v, col.v=3, type="ls", B=FALSE, col=colors
8 [9], lwd=2, scale=TRUE);
9 circos(R=280, cir=db, W=40, mapping=seg.v, col.v=3, type="lh", B=TRUE, col=colors[7],
10 lwd=2, scale=TRUE);
11 circos(R=240, cir=db, W=40, mapping=seg.v, col.v=19, type="ml", B=FALSE, col=colors,
12 lwd=2, scale=TRUE);
```

```

9 | circos(R=200, cir=db, W=40, mapping=seg.v, col.v=19, type="ml2", B=TRUE, col=colors,
   |     lwd=2);
10 | circos(R=160, cir=db, W=40, mapping=seg.v, col.v=19, type="ml3", B=FALSE, cutoff=5,
   |     lwd=2);
11 | circos(R=150, cir=db, W=40, mapping=link.v, type="link", lwd=2, col=colors[c(1,7)]);
12 | 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 1 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.

```

1 | options(stringsAsFactors = FALSE);
2 | library(OmicCircos);
3 | set.seed(1234);
4 |
5 | ## initial values for simulation data
6 | seg.num   <- 10;
7 | ind.num   <- 20;
8 | seg.po    <- c(20:50);

```

```

9 link.num ← 10;
10 link.pg.num ← 4;
11 ## output simulation data
12 sim.out ← sim.circos(seg=seg.num, po=seg.po, ind=ind.num, link=link.num,
13   link.pg=link.pg.num);
14
15 seg.f ← sim.out$seg.frame;
16 seg.v ← sim.out$seg.mapping;
17 link.v ← sim.out$seg.link
18 link.pg.v ← sim.out$seg.link.pg
19 seg.num ← length(unique(seg.f[,1]));
20
21 ## select segments
22 seg.name ← paste("chr", 1:seg.num, sep="");
23 db ← segAnglePo(seg.f, seg=seg.name);
24
25 colors ← rainbow(seg.num, alpha=0.5);

```

```

1 par(mar=c(2, 2, 2, 2));
2 plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="", main="");
3
4 circos(R=400, type="chr", cir=db, col=colors, print.chr.lab=TRUE, W=4, scale=TRUE);
5 circos(R=360, cir=db, W=40, mapping=seg.v, col.v=8, type="box", B=TRUE, col=colors
6   [1], lwd=0.1, scale=TRUE);
7 circos(R=320, cir=db, W=40, mapping=seg.v, col.v=8, type="hist", B=TRUE, col=colors
8   [3], lwd=0.1, scale=TRUE);
9 circos(R=280, cir=db, W=40, mapping=seg.v, col.v=8, type="ms", B=TRUE, col=colors[7],
10   lwd=0.1, scale=TRUE);
11 circos(R=240, cir=db, W=40, mapping=seg.v, col.v=3, type="h", B=FALSE, col=colors
12   [2], lwd=0.1);
13 circos(R=200, cir=db, W=40, mapping=seg.v, col.v=3, type="s", B=TRUE, col=colors, lwd
14   =0.1);
15 circos(R=160, cir=db, W=40, mapping=seg.v, col.v=3, type="b", B=FALSE, col=colors, lwd
16   =0.1);
17 circos(R=150, cir=db, W=40, mapping=link.v, type="link", lwd=2, col=colors[c(1,7)]);
18 circos(R=150, cir=db, W=40, mapping=link.pg.v, type="link.pg", lwd=2, col=sample(
19   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).

```

1 options(stringsAsFactors = FALSE);
2 library(OmicCircos);
3 set.seed(1234);
4
5 ## initial values for simulation data
6 seg.num ← 10;
7 ind.num ← 20;
8 seg.po ← c(20:50);
9 link.num ← 10;
10 link.pg.num ← 4;
11 ## output simulation data
12 sim.out ← sim.circos(seg=seg.num, po=seg.po, ind=ind.num, link=link.num,
13   link.pg=link.pg.num);

```



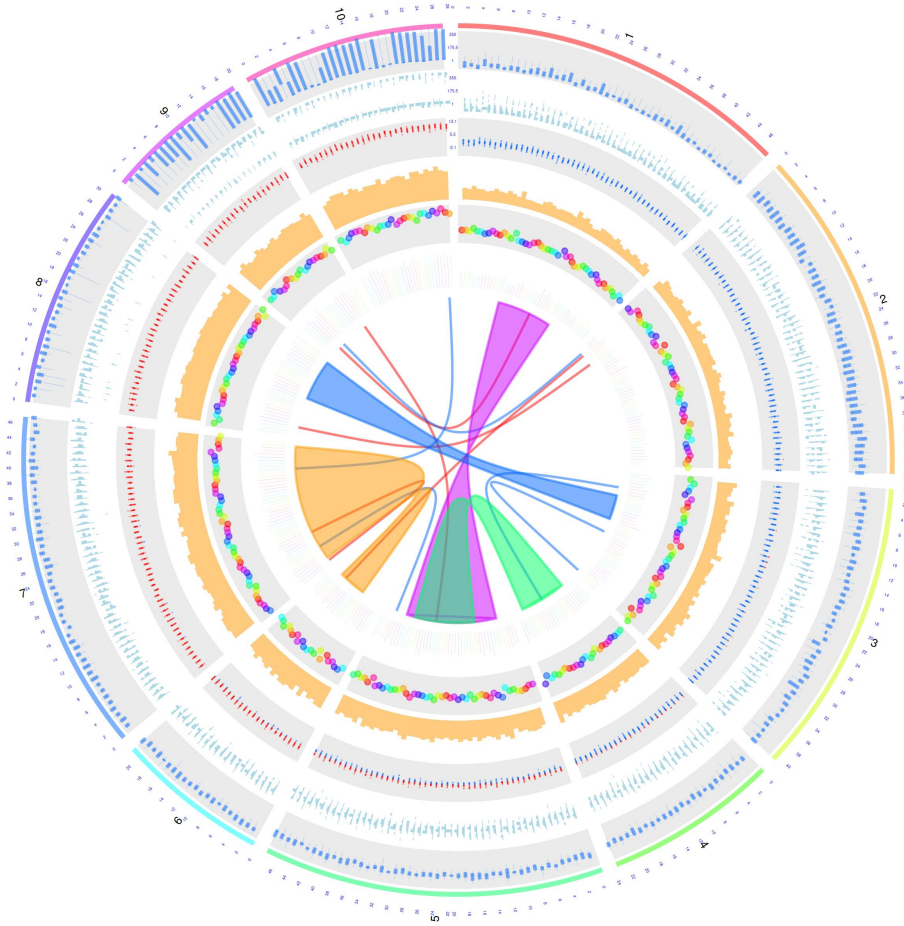


Figure 2

```

14
15 seg.f      ← sim.out$seg.frame;
16 seg.v      ← sim.out$seg.mapping;
17 link.v     ← sim.out$seg.link
18 link.pg.v  ← sim.out$seg.link.pg
19 seg.num    ← length(unique(seg.f[,1]));
20
21 ##
22 seg.name   ← paste("chr", 1:seg.num, sep="");
23 db        ← segAnglePo(seg.f, seg=seg.name);
24
25 colors    ← rainbow(seg.num, alpha=0.5);

```

```

1 par(mar=c(2, 2, 2, 2));
2 plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="", main="");
3
4 circos(R=400, type="chr", cir=db, col=colors, print.chr.lab=TRUE, W=4, scale=TRUE);
5 circos(R=360, cir=db, W=40, mapping=seg.v, col.v=8, type="quant90", B=FALSE, col=
  colors, lwd=2, scale=TRUE);
6 circos(R=320, cir=db, W=40, mapping=seg.v, col.v=3, type="sv", B=TRUE, col=colors[7],
  scale=TRUE);
7 circos(R=280, cir=db, W=40, mapping=seg.v, col.v=3, type="ss", B=FALSE, col=colors[3],
  scale=TRUE);
8 circos(R=240, cir=db, W=40, mapping=seg.v, col.v=8, type="heatmap", lwd=3);
9 circos(R=200, cir=db, W=40, mapping=seg.v, col.v=3, type="s.sd", B=FALSE, col=colors
  [4]);
10 circos(R=160, cir=db, W=40, mapping=seg.v, col.v=3, type="ci95", B=TRUE, col=colors
  [4], lwd=2);
11 circos(R=150, cir=db, W=40, mapping=link.v, type="link", lwd=2, col=colors[c(1,7)]);
12 circos(R=150, cir=db, W=40, mapping=link.pg.v, type="link.pg", lwd=2, col=sample(
  colors, link.pg.num));
13
14 the.col1=rainbow(10, alpha=0.5)[3];
15 highlight ← c(160, 410, 6, 2, 6, 10, the.col1, the.col1);
16 circos(R=110, cir=db, W=40, mapping=highlight, type="hl", lwd=1);
17
18 the.col1=rainbow(10, alpha=0.1)[3];
19 the.col2=rainbow(10, alpha=0.5)[1];
20 highlight ← c(160, 410, 3, 12, 3, 20, the.col1, the.col2);
21 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 radius=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 interval of the samples.

## 4.2 annotation

```

1 options(stringsAsFactors = FALSE);
2 library(OmicCircos);
3 set.seed(1234);

```

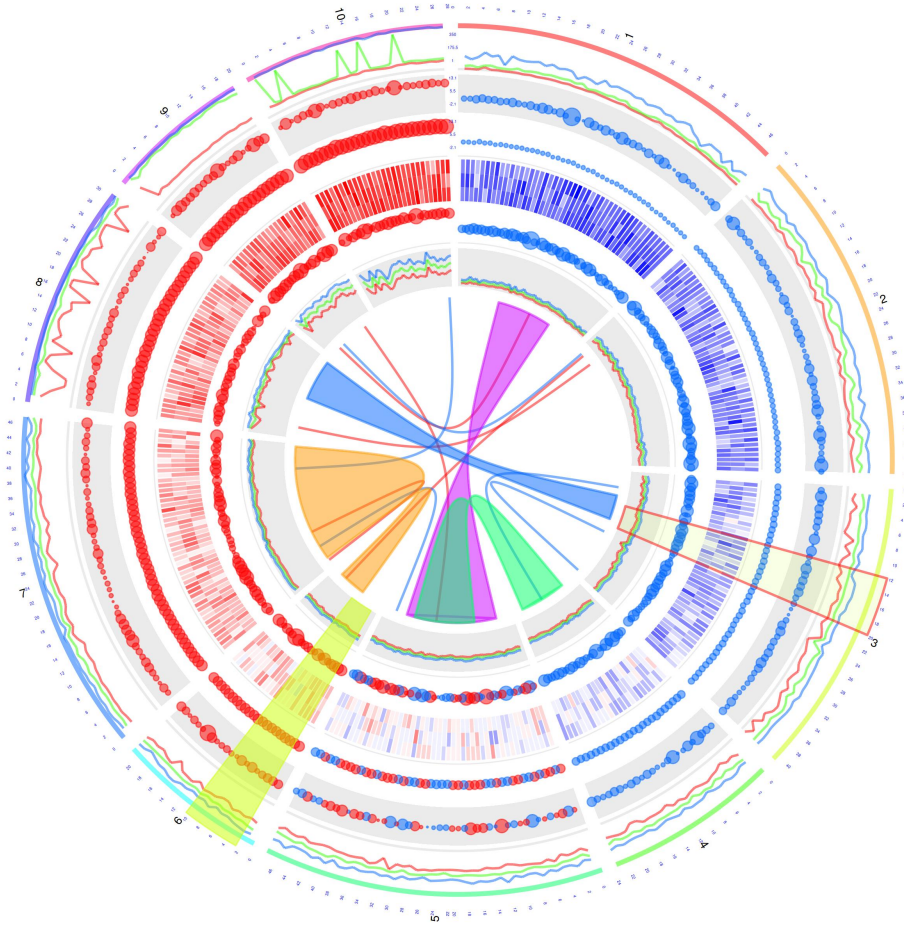


Figure 3

```

4
5 ## load mm cytogenetic band data
6 data("UCSC.mm10.chr", package="OmicCircos");
7 ref      ← UCSC.mm10.chr;
8 ref[,1] ← gsub("chr", "", ref[,1]);
9 ## initial values for simulation data
10 colors ← rainbow(10, alpha=0.8);
11 lab.n  ← 50;
12 cnv.n  ← 200;
13 arc.n  ← 30;
14 fus.n  ← 10;
15
16 ## make arc data
17 arc.d ← c();
18 for (i in 1:arc.n){
19   chr      ← sample(1:19, 1);
20   chr.i    ← which(ref[,1]==chr);
21   chr.arc  ← ref[chr.i,];
22   arc.i    ← sample(1:nrow(chr.arc), 2);
23   arc.d    ← rbind(arc.d, c(chr.arc[arc.i[1],c(1,2)], chr.arc[arc.i[2],c(2,4)]));
24 }
25 colnames(arc.d) ← c("chr", "start", "end", "value");
26
27 ## make fusion
28 fus.d ← c();
29 for (i in 1:fus.n){
30   chr1     ← sample(1:19, 1);
31   chr2     ← sample(1:19, 1);
32   chr1.i   ← which(ref[,1]==chr1);
33   chr2.i   ← which(ref[,1]==chr2);
34   chr1.f   ← ref[chr1.i,];
35   chr2.f   ← ref[chr2.i,];
36   fus1.i   ← sample(1:nrow(chr1.f), 1);
37   fus2.i   ← sample(1:nrow(chr2.f), 1);
38   n1       ← paste0("geneA", i);
39   n2       ← paste0("geneB", i);
40   fus.d    ← rbind(fus.d, c(chr1.f[fus1.i,c(1,2)], n1, chr2.f[fus2.i,c(1,2)], n2));
41 }
42 colnames(fus.d) ← c("chr1", "pol", "gene1", "chr2", "po2", "gene2");
43
44 cnv.i ← sample(1:nrow(ref), cnv.n);
45 vale  ← rnorm(cnv.n);
46 cnv.d ← data.frame(ref[cnv.i,c(1,2)], value=vale);

```

```

1 par(mar=c(2, 2, 2, 2));
2 plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="");
3
4 circos(R=400, type="chr", cir="mm10", print.chr.lab=TRUE, W=4, scale=TRUE);
5 circos(R=340, cir="mm10", W=60, mapping=cnv.d, type="b3", B=TRUE, col=colors[7]);
6 circos(R=340, cir="mm10", W=60, mapping=cnv.d, type="s2", B=FALSE, col=colors[1], cex
  =0.5);
7 circos(R=280, cir="mm10", W=60, mapping=arc.d, type="arc2", B=FALSE, col=colors, lwd
  =10, cutoff=0);
8 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);

```

```

9 | 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);
10 | circos(R=150, cir="mm10", W=10, mapping=fus.d, type="link", lwd=2, col=colors[c
    | (1,7,9)]);

```

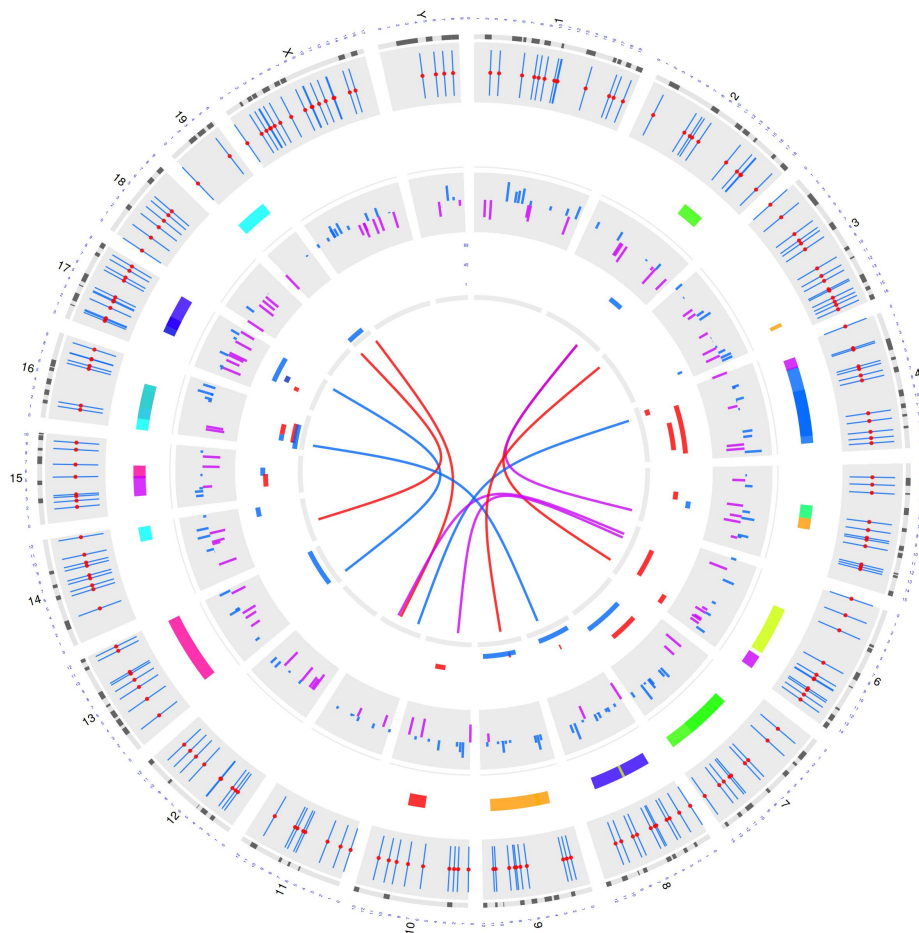


Figure 4

### 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.

```

1 | options(stringsAsFactors = FALSE);
2 | library(OmicCircos);
3 |
4 | data("TCGA.PAM50_genefu_hg18");
5 | data("TCGA.BC.fus");
6 | data("TCGA.BC.cnv.2k.60");

```

```

7 data ("TCGA.BC.gene.exp.2k.60");
8 data ("TCGA.BC.sample60");
9 data ("TCGA.BC_Her2_cnv_exp");
10
11 pvalue <- -1 * log10(TCGA.BC_Her2_cnv_exp[,5]);
12 pvalue <- cbind(TCGA.BC_Her2_cnv_exp[,c(1:3)], pvalue);
13
14 Her2.i <- which(TCGA.BC.sample60[,2] == "Her2");
15 Her2.n <- TCGA.BC.sample60[Her2.i,1];
16
17 Her2.j <- which(colnames(TCGA.BC.cnv.2k.60) %in% Her2.n);
18 cnv <- TCGA.BC.cnv.2k.60[,c(1:3,Her2.j)];
19 cnv.m <- cnv[,c(4:ncol(cnv))];
20 cnv.m[cnv.m > 2] <- 2;
21 cnv.m[cnv.m < -2] <- -2;
22 cnv <- cbind(cnv[,1:3], cnv.m);
23
24 Her2.j <- which(colnames(TCGA.BC.gene.exp.2k.60) %in% Her2.n);
25 gene.exp <- TCGA.BC.gene.exp.2k.60[,c(1:3,Her2.j)];
26 colors <- rainbow(10, alpha=0.5);

1 par(mar=c(2, 2, 2, 2));
2 plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="");
3
4 circos(R=300, type="chr", cir="hg18", print.chr.lab=FALSE, W=4);
5 circos(R=310, cir="hg18", W=20, mapping=TCGA.PAM50_genefu_hg18, type="label",
6       side="out", col=c("black", "blue", "red"), cex=0.4);
7 circos(R=250, cir="hg18", W=50, mapping=cnv, col.v=4, type="ml3", B=FALSE, col=colors
8       [7], cutoff=0, scale=TRUE);
9 circos(R=200, cir="hg18", W=50, mapping=gene.exp, col.v=4, type="ml3", B=TRUE, col=
10      colors[3], cutoff=0, scale=TRUE);
11 circos(R=140, cir="hg18", W=50, mapping=pvalue, col.v=4, type="l", B=FALSE, col=colors
12      [1], scale=TRUE);
13 ## set fusion gene colors
14 cols <- rep(colors[7], nrow(TCGA.BC.fus));
15 col.i <- which(TCGA.BC.fus[,1]==TCGA.BC.fus[,4]);
16 cols[col.i] <- colors[1];
17 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.

```

1 par(mar=c(2, 2, 2, 2));
2 plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="", main="");
3 circos(R=300, type="chr", cir="hg18", col=TRUE, print.chr.lab=FALSE, W=4);
4 circos(R=290, cir="hg18", W=20, mapping=TCGA.PAM50_genefu_hg18, type="label", side="in
5       ", col=c("black", "blue"), cex=0.4);
6 circos(R=310, cir="hg18", W=50, mapping=cnv, col.v=4, type="ml3", B=TRUE, col=colors
7       [7], cutoff=0, scale=TRUE);
8 circos(R=150, cir="hg18", W=50, mapping=gene.exp, col.v=4, type="ml3", B=TRUE, col=
9       colors[3], cutoff=0, scale=TRUE);
10 circos(R=90, cir="hg18", W=50, mapping=pvalue, col.v=4, type="l", B=FALSE, col=colors
11      [1], scale=TRUE);
12 circos(R=82, cir="hg18", W=50, mapping=TCGA.BC.fus, type="link", col=cols, lwd=2);

```

Figure 6 is an example of the inside labels.

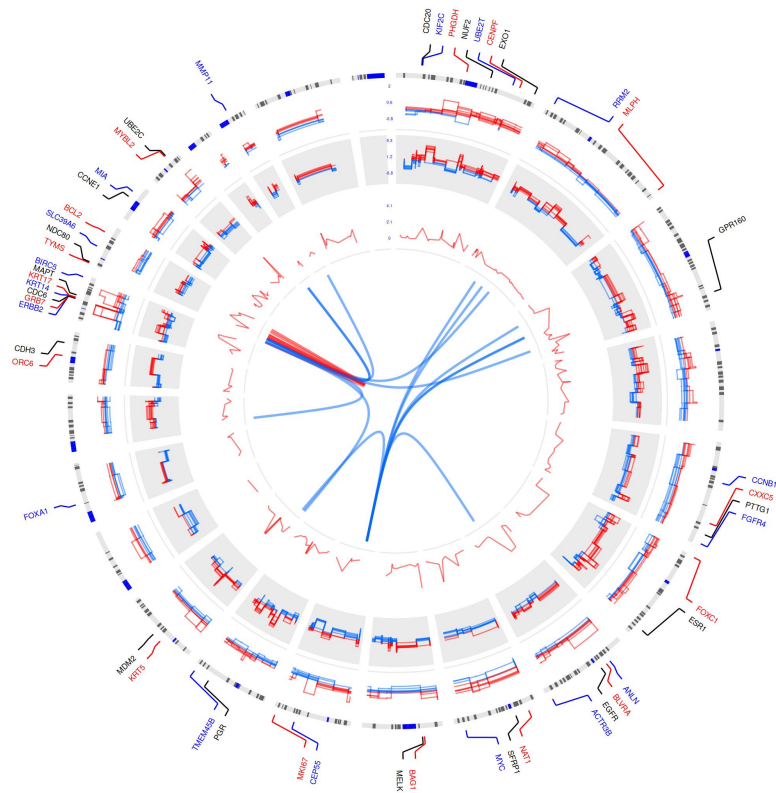


Figure 5

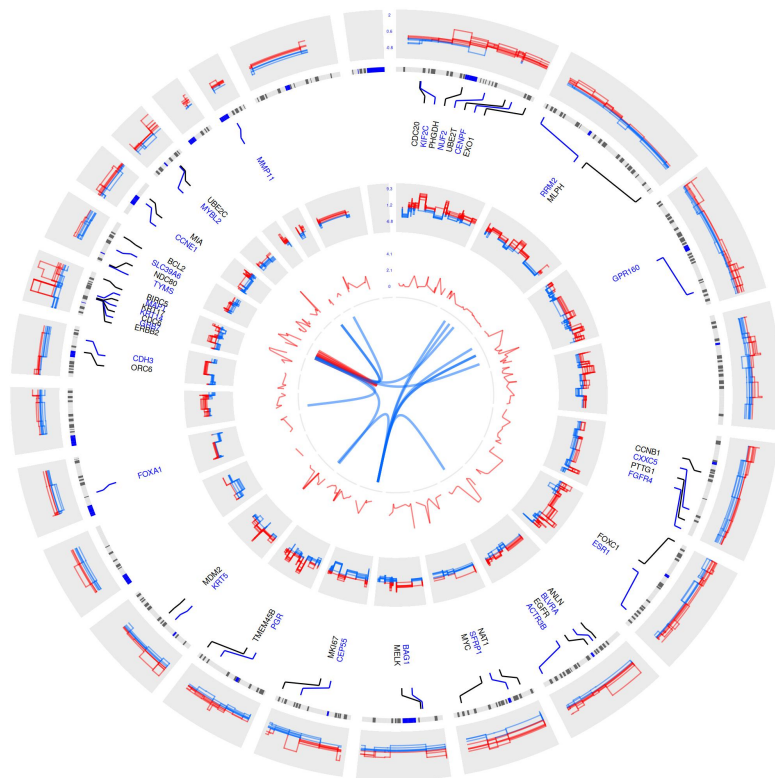


Figure 6

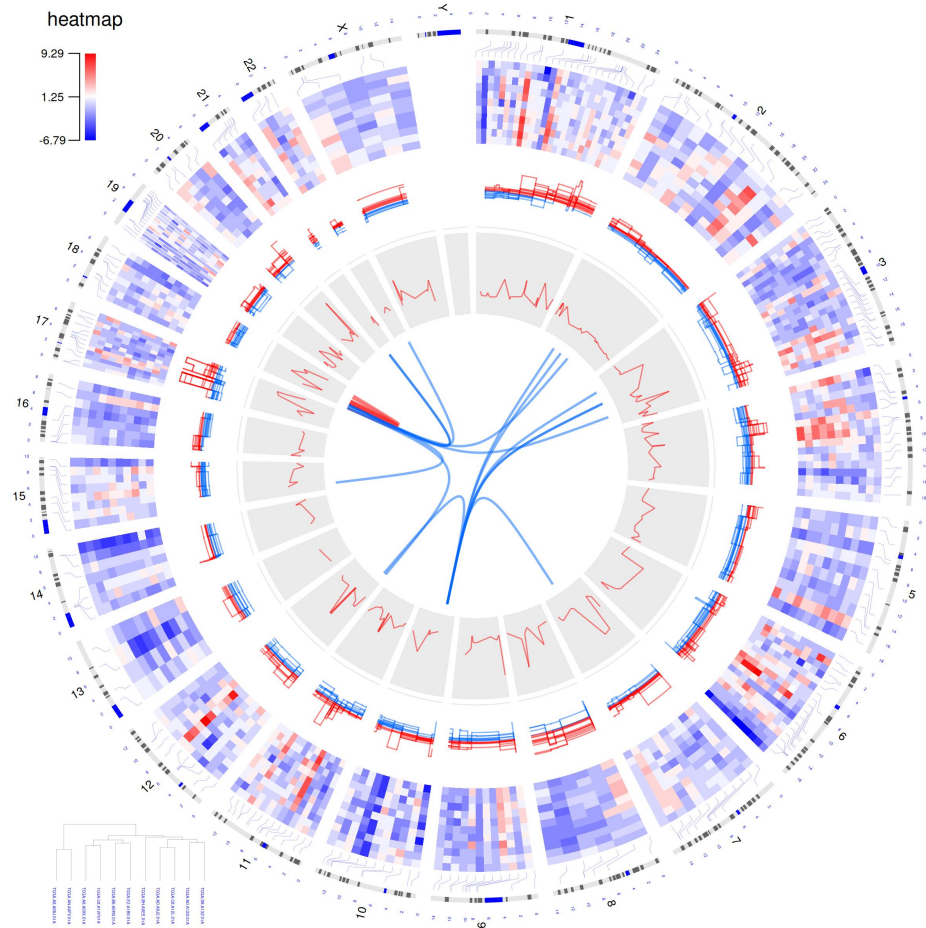


## 4.4 heatmap

```
1 options(stringsAsFactors = FALSE);
2 library(OmicCircos);
3
4 data("TCGA.PAM50_genefu_hg18");
5 data("TCGA.BC.fus");
6 data("TCGA.BC.cnv.2k.60");
7 data("TCGA.BC.gene.exp.2k.60");
8 data("TCGA.BC.sample60");
9 data("TCGA.BC_Her2_cnv_exp");
10
11 pvalue <- -1 * log10(TCGA.BC_Her2_cnv_exp[,5]);
12 pvalue <- cbind(TCGA.BC_Her2_cnv_exp[,c(1:3)], pvalue);
13
14 Her2.i <- which(TCGA.BC.sample60[,2] == "Her2");
15 Her2.n <- TCGA.BC.sample60[Her2.i,1];
16
17 Her2.j <- which(colnames(TCGA.BC.cnv.2k.60) %in% Her2.n);
18 cnv <- TCGA.BC.cnv.2k.60[,c(1:3, Her2.j)];
19 cnv.m <- cnv[,c(4:ncol(cnv))];
20 cnv.m[cnv.m > 2] <- 2;
21 cnv.m[cnv.m < -2] <- -2;
22 cnv <- cbind(cnv[,1:3], cnv.m);
23
24 Her2.j <- which(colnames(TCGA.BC.gene.exp.2k.60) %in% Her2.n);
25 gene.exp <- TCGA.BC.gene.exp.2k.60[,c(1:3, Her2.j)];
26
27 colors <- rainbow(10, alpha=0.5);

1 par(mar=c(2, 2, 2, 2));
2
3 plot(c(1,800), c(1,800), type="n", axes=FALSE, xlab="", ylab="", main="");
4
5 circos(R=400, cir="hg18", W=4, type="chr", print.chr.lab=TRUE, scale=TRUE);
6 circos(R=300, cir="hg18", W=100, mapping=gene.exp, col.v=4, type="heatmap2",
7       cluster=TRUE, col.bar=TRUE, lwd=0.1, col="blue");
8 circos(R=220, cir="hg18", W=80, mapping=cnv, col.v=4, type="ml3", B=FALSE, lwd=1,
9       cutoff=0);
10 circos(R=140, cir="hg18", W=80, mapping=pvalue, col.v=4, type="l", B=TRUE, lwd
11       =1, col=colors[1]);
12
13 cols <- rep(colors[7], nrow(TCGA.BC.fus));
14 col.i <- which(TCGA.BC.fus[,1]==TCGA.BC.fus[,4]);
15 cols[col.i] <- colors[1];
16 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**