Customizing RCircos Plot: How to

Hongen Zhang, Ph.D.
hzhang@mail.nih.gov
Genetics Branch, Center for Cancer Research,
National Cancer Institute, NIH
Bethesda, Maryland, USA

November 28, 2013

Contents

1 Introduction

RCircos package [1], developed with R graphics packages, implements basic Circos 2D track plot [2]. The RCircos itself is a set of graphic and supporting functions that are not only easy to use to generate basic Circos 2D track plot images but also flexible for customizing the plots. The vignette of RCircos package provids a short guide for how to make basic RCircos plots and this document is focused on how to customize the RCircos plots.

RCircos uses three core components to control and implement the plots. Each component is supplied with a GET and a RESET function:

- RCircos plot parameters: RCircos.PlotPar
    - GET function: RCircos.Get.Plot.Parameters()
    - RESET function: RCircos.Reset.Plot.Parameters(new.parames)
- RCircos plot ideograms: RCircos.Cytoband
    - GET function: RCircos.Get.Plot.Ideogram()
    - RESET function: RCircos.Reset.Plot.Ideogram(new.ideogram)
- RCircos plot positions: RCircos.Base.Position
    - GET function: RCircos.Get.Plot.Positions()
    - RESET function: RCircos.Reset.Plot.Positions(new.positions)
      Note: Version 1.1.3 or above only.

These three components are initialized after call to the function of RCircos.Set.Core.Components(). By modifying these components, users can simply customize their plots or write their own plot functions to meet specific requirements.

```
> #        Load RCircos library and chromosome ideogram data
> #        ==================================================
> library(RCircos);
> data(UCSC.HG19.Human.CytoBandIdeogram);
> human.cyto <- UCSC.HG19.Human.CytoBandIdeogram;
> #
> #        Initialize RCircos core components
> #        =================================================
> RCircos.Set.Core.Components(human.cyto, chr.exclude=NULL,
+         tracks.inside=5, tracks.outside=0);
```

```
RCircos.Core.Components initialized.
Type ?RCircos.Reset.Plot.Parameters to see how to modify the core components.
```

2 Working with RCircos.PlotPar Object

2.1 The RCircos.PlotPar Object

RCircos plot parameters could be checked with RCircos.List.Parameters() function.

```
> RCircos.List.Parameters();
```

```
Parameters for current RCircos session.

Parameters relative to radius.len:
radius.len:         1.24
chr.ideog.pos:         1.34
highlight.pos:         1.49
chr.name.pos:         1.64
plot.radius:         1.74
track.in.start:         1.29
track.out.start: 1.84
chrom.width:         0.1
track.padding:         0.02
track.height:         0.1

Parameters relative to chromosome unit:
base.per.unit:         3000
chrom.paddings:         3000
heatmap.width:         1000
hist.width:         1000

General R graphic parameters:
text.size:         0.4
highlight.width: 1
point.type:         .
point.size:         1
text.color         black
heatmap.color         BlueWhiteRed
hist.color:         red
line.color:         black
scatter.color:         black
tile.color:         black
track.background:         wheat
grid.line.color:         gray
Bezier.point:         1000
max.layers:         5
sub.tracks:         5

Following are procedures to change RCircos plot parameters:

params <- RCircos.Get.Plot.Parameters();
params$radius.len <- 2.0;
params$base.per.unit <- 5000;
RCircos.Reset.Plot.Parameters(params)

Chromosome ideogram data were automatically modified.
```

There are three types of plot parameters:

- Parameters relative to radius.len

  These parameters are calculated based on the radius.len, the radius of the circle which holds data tracks. Chromosome ideogram will be outside of this circle. Reset radiu.len will cause other related parameters changed automatically.

  ```
  > #        Change radius.len
  > #        ================================================
  > params <- RCircos.Get.Plot.Parameters();
  > params$radius.len <- 2;
  > RCircos.Reset.Plot.Parameters(params);
  > RCircos.List.Parameters();

  Parameters for current RCircos session.

  Parameters relative to radius.len:
  radius.len:         2
  chr.ideog.pos:         2.1
  highlight.pos:         2.25
  chr.name.pos:         2.4
  plot.radius:         2.5
  track.in.start:         2.05
  track.out.start: 2.5
  chrom.width:         0.1
  track.padding:         0.02
  track.height:         0.1

  Parameters relative to chromosome unit:
  base.per.unit:         3000
  chrom.paddings:         3000
  heatmap.width:         1000
  hist.width:         1000

  General R graphic parameters:
  text.size:         0.4
  highlight.width: 2
  point.type:         .
  point.size:         1
  text.color         black
  heatmap.color         BlueWhiteRed
  hist.color:         red
  line.color:         black
  scatter.color:         black
  ```

```
tile.color:         black
track.background:         wheat
grid.line.color:         gray
Bezier.point:         1000
max.layers:         5
sub.tracks:         5

Following are procedures to change RCircos plot parameters:

params <- RCircos.Get.Plot.Parameters();
params$radius.len <- 2.0;
params$base.per.unit <- 5000;
RCircos.Reset.Plot.Parameters(params)

Chromosome ideogram data were automatically modified.
```

- Parameters relative to chromosome unit

  The term of chromosome unit is addopted from Circos software and in RCircos it represents a point on image plot. By default, a chromosome unit will represent 3000 base pairs. For human genomes, total chromosome units are 1,103,905. Increasing values of base.per.unit will reduce the number of points to be plotted and make plot running fast but also reduce the image resolution. In general, it is recommended not to change this value.

  The values of heatmap.width and hist.width are how width of a data point will be plotted on the track. They could be increased to make the plot more readable if the total number of data points is small.

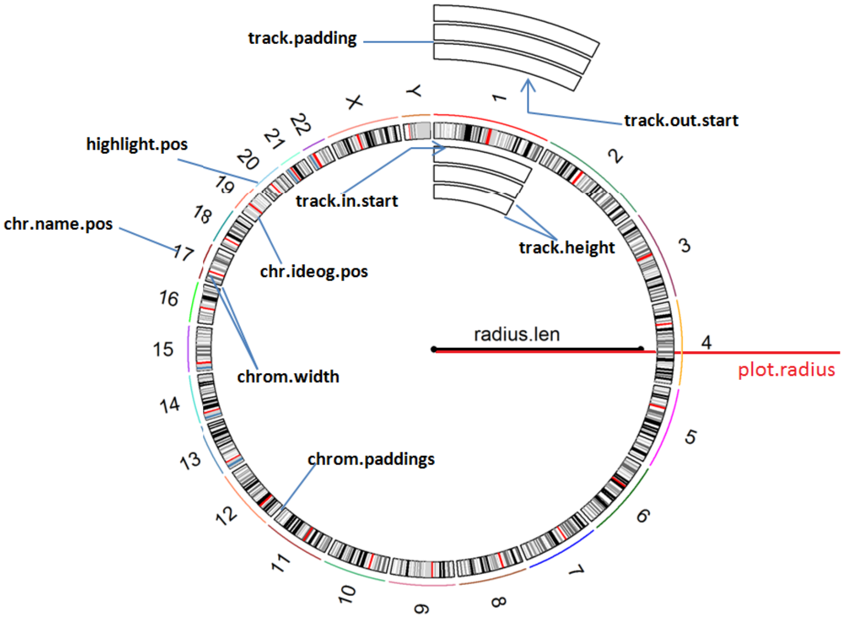
- General R graphic parameters

  The parameters in this catagory are all common R graphic parameters, either float number, integer, or character vectors.

  Parameters with character values: color names
  text.size: used for cex in text plot
  highlight.width: used for lwd (line width)
  point.type: used for pch in scatter plot
  point.size: used for cex in scatter plot
  Bezier.point: total number of points for a Bezier line
  max.layers: number of sub layer for tile plot
  sub.tracks: number of lines to mark the sub-track of a data track

2.2 Customize Plot Layout

The figure below shows parameters related to the layout of RCircos Plot. Customizing RCircos plot layout is mainly done by modifying the parameters related to radius.len and chromosome unit.



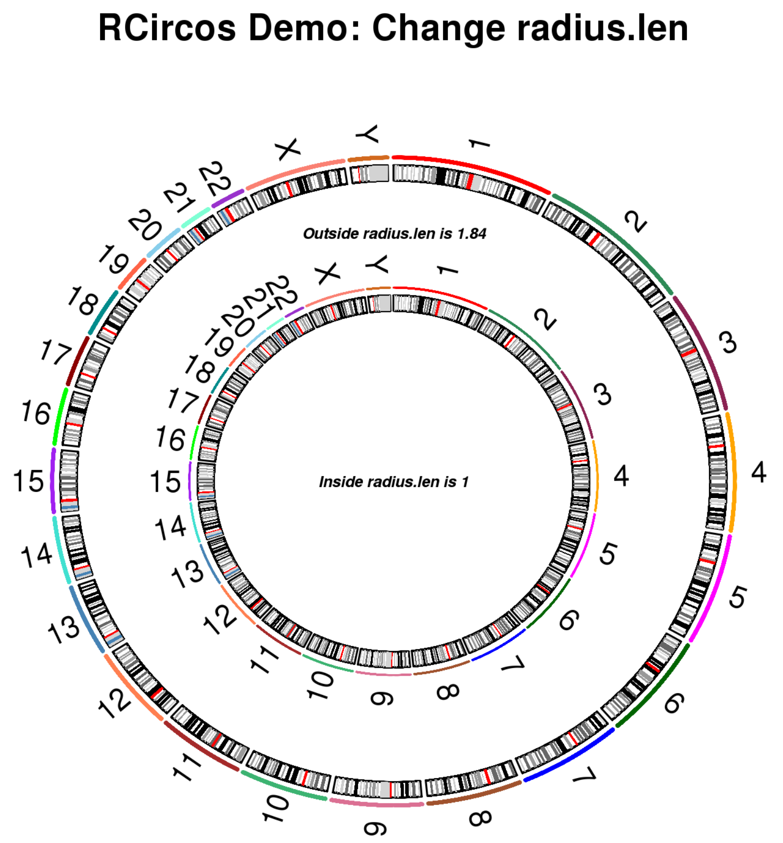
Following sample code show the procedures and outputs of changing radius length, track height, and heatmap width.

- Change the Circle Radius

  The values of radius.len decides the plot area inside of chromosome ideogram. Increase radius.len give more room to plot more data tracks and reduce the radius.len will allow less data tracks inside chromosome ideogram. Note: the best way to obtain desired plot area is to call the function of RCircos.Set.Core.Components() with correct track numbers of inside and outside of chromosome ideogram. Only modify the radius.len in case reduce radius.len is necessary.

  Following code sample will generate an image with two sets of chromosome ideograms, one with orignal radius.len and anothere with reduced radius.len value.

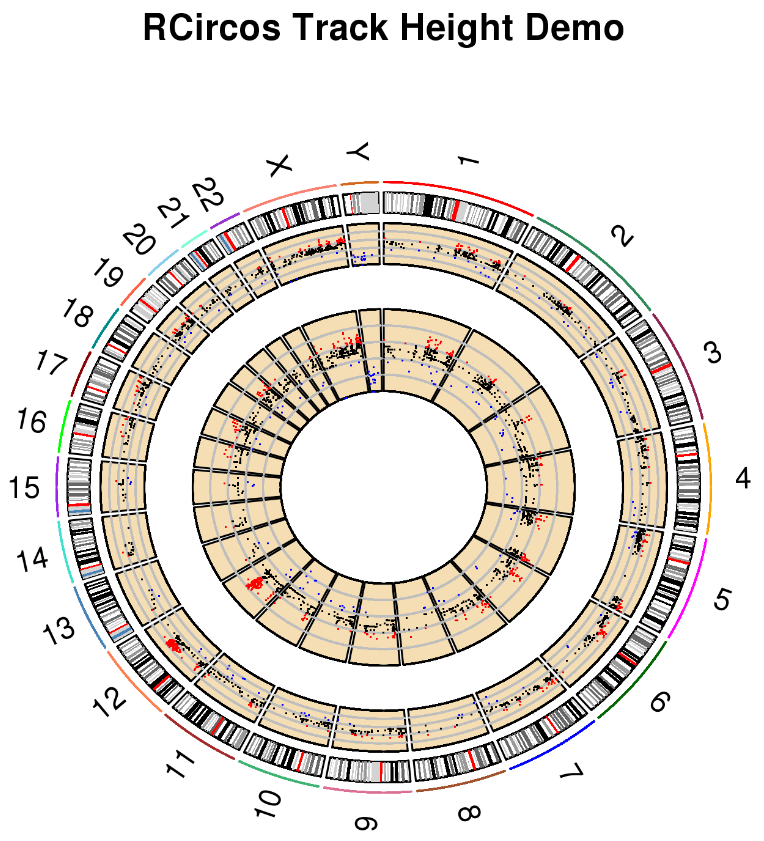
```
> #
> #        Load RCircos library and chromosome ideogram data
> #        ================================================
> library(RCircos);
> data(UCSC.HG19.Human.CytoBandIdeogram);
> human.cyto <- UCSC.HG19.Human.CytoBandIdeogram;
> #
> #        Initialize RCircos core components
> #        ===============================================
> RCircos.Set.Core.Components(human.cyto, chr.exclude=NULL,
+         tracks.inside=10, tracks.outside=0);
> #
> #        Plot chromosome ideogram with original radius
> #        ================================================
> RCircos.Set.Plot.Area();
> RCircos.Chromosome.Ideogram.Plot();
> params <- RCircos.Get.Plot.Parameters();
> note <- paste("Outside radius.len is", params$radius.len);
> text(0, 1.6, note, font=4);
> #
> #          Reduce radius length to 1.0
> #        ================================================
> params$radius.len <- 1;
> RCircos.Reset.Plot.Parameters(params);
> RCircos.Chromosome.Ideogram.Plot();
> note <- paste("Inside radius.len is", params$radius.len);
> text(0, 0, note, font=4);
> title("RCircos Demo: Change radius.len");
```



- Change Track Height

  Parameters for data tracks such as track height and track padding could be modified before data plot. Following code show two sactter plot tracks with different track heights.

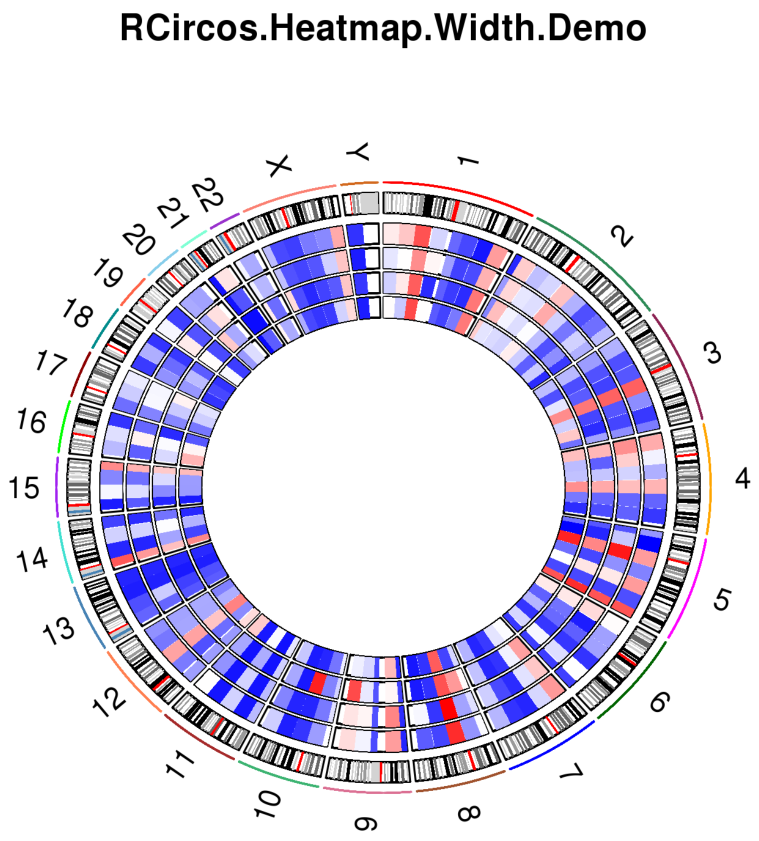
```
> #
> #        Load RCircos library and chromosome ideogram data
> #        ==================================================
> library(RCircos);
> data(UCSC.HG19.Human.CytoBandIdeogram);
> human.cyto <- UCSC.HG19.Human.CytoBandIdeogram;
> #
> #        Initialize RCircos core components
> #        =================================================
> RCircos.Set.Core.Components(human.cyto, chr.exclude=NULL,
+         tracks.inside=5, tracks.outside=0);
> #
> #        Plot chromosome ideogram
> #        ================================================
> RCircos.Set.Plot.Area();
> RCircos.Chromosome.Ideogram.Plot();
> #
> #        Scatterplot with default track height
> #        ========================================
> data(RCircos.Scatter.Data);
> RCircos.Scatter.Plot(RCircos.Scatter.Data,
+                 data.col=5, track.num=1,
+                 side="in", by.fold=1);
> #
> #          Reset (double) the track height
> #          ========================================
> params <- RCircos.Get.Plot.Parameters();
> params$track.height <- params$track.height*2;
> RCircos.Reset.Plot.Parameters(params);
> #
> #        Scatterplot with new track height
> #        ========================================
> RCircos.Scatter.Plot(RCircos.Scatter.Data,
+                 data.col=5, track.num=2,
+                 side="in", by.fold=1);
> title("RCircos Track Height Demo");
```



- Change Heatmap Width

  The default width of a heatmap cell is 1000 chromosome units. This makes the heatmap cells like colored vertical lines inside of a data track. When there are only small number of data to be plotted, the heatmap cell width could be increased to make the heatmap more clear. This is done by modifying the heatmap.width parameter.

```
> #
> #        Load RCircos library and chromosome ideogram data
> #        ================================================
> library(RCircos);
> data(UCSC.HG19.Human.CytoBandIdeogram);
> human.cyto <- UCSC.HG19.Human.CytoBandIdeogram;
> #
> #        Initialize RCircos core components
> #        ================================================
> RCircos.Set.Core.Components(human.cyto, chr.exclude=NULL,
+         tracks.inside=5, tracks.outside=0);
> #
> #          Reset heatmap cell width
> #          ================================================
> params <- RCircos.Get.Plot.Parameters()
> params$heatmap.width <- 10000;
> RCircos.Reset.Plot.Parameters(params)
> #
> #        Plot chromosome ideogram
> #        ================================================
> RCircos.Set.Plot.Area();
> RCircos.Chromosome.Ideogram.Plot();
> #
> #        Reduce the size of heatmap data
> #        ================================================
> data(RCircos.Heatmap.Data);
> heatmap.data <- RCircos.Get.Plot.Data(RCircos.Heatmap.Data,
+         plot.type="plot");
> selected <- round(heatmap.data$Location/10000, digits=0);
> heatmap.data <- heatmap.data[duplicated(selected)==F,];
> #
> #        Plot four tracks of heapmap
> #        ================================================
> side <- "in";
> for(track.num in 1:4){
+         data.col <- track.num + 4;
+         RCircos.Heatmap.Plot(heatmap.data, data.col,
+                 track.num, side)
+ }
> #
> title("RCircos.Heatmap.Width.Demo");
```



2.3 Customize Heatmap Plot Colors

Color map for heatmap plot is controlled by heatmap.color parameter. There are total of six color maps supported by RCircos heatmap plot.

BlueWhiteRed color map ranges from pure blue at low end, through white in the middle, to pure red at the high end.

GreenWhiteRed color map ranges from pure green at low end, through white in the middle, to pure red at the high end.

GreenYellowRed color map ranges from pure green at low end, through yellow in the middle, to pure red at the high end.

GreenBlackRed color map ranges from pure green at low end, through black in the middle, to pure red at the high end.

YellowToRed color map ranges from yellow at low end to red at the high end.

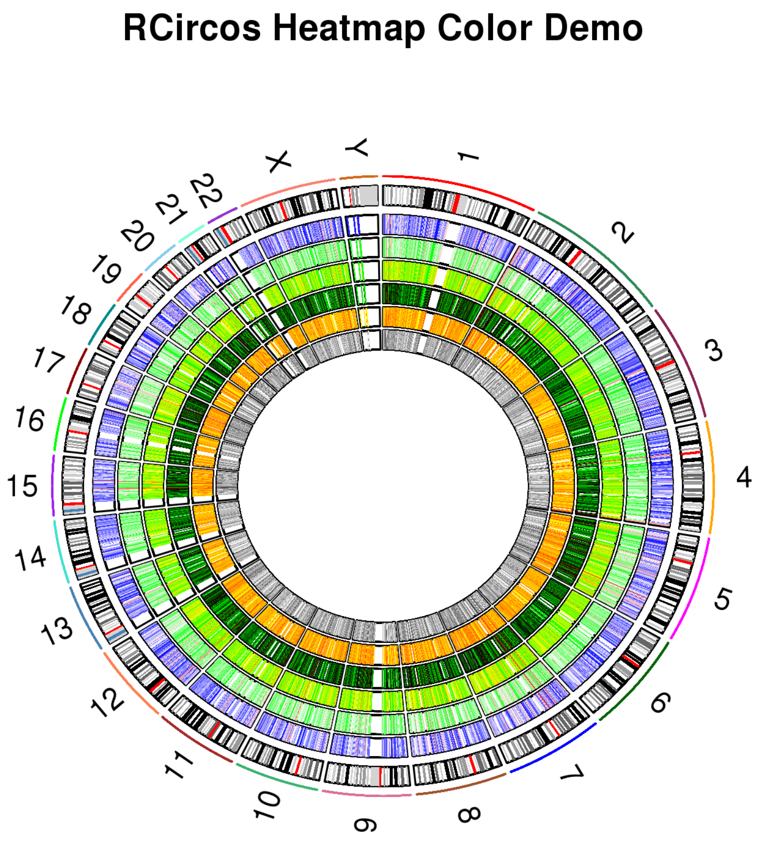
BlackOnly color map ranges from white at low end to red at the high end.

Users can change the heatmap color by modifying heatmap.color parameter. A typical procedure to change heatmap color is as following:

```
> params <- RCircos.Get.Plot.Parameters();
> params$heatmap.color <- "GreenWhiteRed";
> RCircos.Reset.Plot.Parameters(params);
```

The code below will generate an heatmap image with different color maps.

```
> #
> #        Load RCircos library and chromosome ideogram data
> #        ==================================================
> library(RCircos);
> data(UCSC.HG19.Human.CytoBandIdeogram);
> human.cyto <- UCSC.HG19.Human.CytoBandIdeogram;
> #
> #        Initialize RCircos core components
> #        =================================================
> RCircos.Set.Core.Components(human.cyto, chr.exclude=NULL,
+         tracks.inside=6, tracks.outside=0);
> #
> #        Plot chromosome ideogram
> #        =================================================
> RCircos.Set.Plot.Area();
> RCircos.Chromosome.Ideogram.Plot()
> #
> data(RCircos.Heatmap.Data);
> #
> #        Heatmap plot with default color map (BlueWhiteRed)
> #        =================================================
> RCircos.Heatmap.Plot(RCircos.Heatmap.Data, data.col=6,
+                 track.num=1, side="in");
> #
> #          Heatmap plot with GreenWhiteRed colors
> #          =================================================
> params <- RCircos.Get.Plot.Parameters();
> params$heatmap.color <- "GreenWhiteRed";
> RCircos.Reset.Plot.Parameters(params);
> RCircos.Heatmap.Plot(RCircos.Heatmap.Data, data.col=6,
+                 track.num=2, side="in");
> #
> #          Heatmap plot with GreenYellowRed colors
> #          =================================================
> params$heatmap.color <- "GreenYellowRed"
> RCircos.Reset.Plot.Parameters(params)
> RCircos.Heatmap.Plot(RCircos.Heatmap.Data, data.col=6,
+                 track.num=3, side="in");
> #
> #          Heatmap plot with GreenBlackRed colors
> #          ================================================
> params$heatmap.color <- "GreenBlackRed"
> RCircos.Reset.Plot.Parameters(params)
> RCircos.Heatmap.Plot(RCircos.Heatmap.Data, data.col=6,
+                 track.num=4, side="in");
> #
> #          Heatmap plot with YellowToRed colors
> #          ================================================
> params$heatmap.color <- "YellowToRed"
> RCircos.Reset.Plot.Parameters(params)
> RCircos.Heatmap.Plot(RCircos.Heatmap.Data, data.col=6,
+                 track.num=5, side="in");
> #
> #          Heatmap plot with Black color only
> #          ================================================
> params$heatmap.color <- "BlackOnly"
> RCircos.Reset.Plot.Parameters(params)
> RCircos.Heatmap.Plot(RCircos.Heatmap.Data, data.col=6,
+                 track.num=6, side="in");
> #
> title("RCircos Heatmap Color Demo")
```

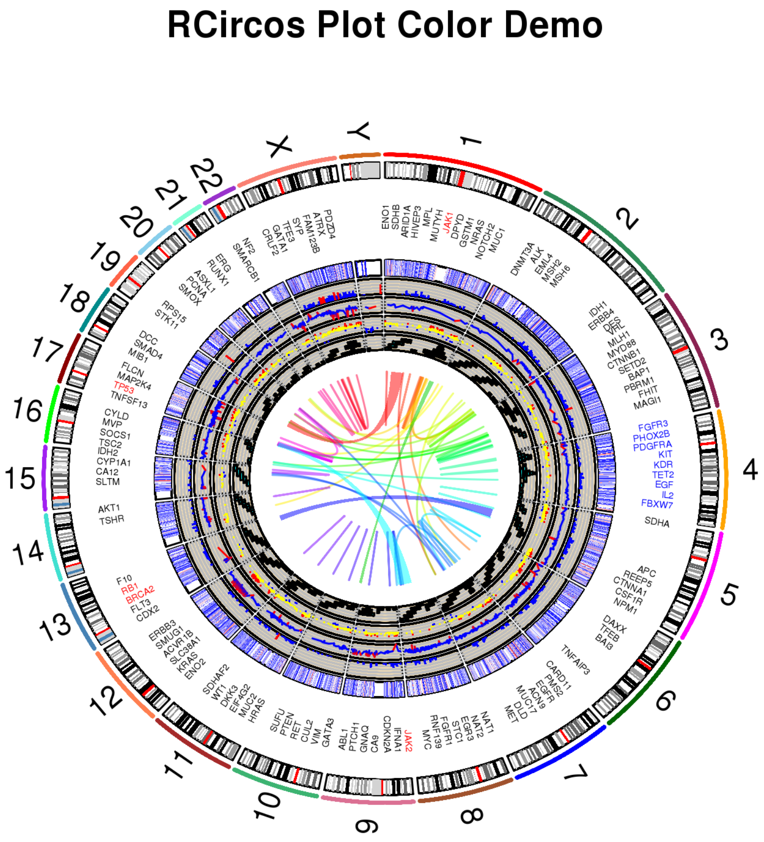


2.4 Customize Other Plot Colors

Default colors for other plot items such as text, scatter, histogram, lines, ... are also defined in the RCircos.PlotPar object. When a new color is desired for all data points in one track, users can modify the relavent color for that plot.

Starting from version 1.1.2, RCircos provides another flexible way to customize plot colors except of heatmap color, i.e., using one extra column in input dataset to define colors for each data point (row). This column could physically exist in data file or be generated and appended to input data during R session but its column header must be ”PlotColor”. In the example below default colors have been changed for all data tracks except of heatmap.

```
> #
> #        Load RCircos library and chromosome ideogram data
> #        ================================================
> library(RCircos);
> data(UCSC.HG19.Human.CytoBandIdeogram);
> human.cyto <- UCSC.HG19.Human.CytoBandIdeogram;
> #
> #        Initialize RCircos core components
> #        ================================================
> RCircos.Set.Core.Components(human.cyto, chr.exclude=NULL,
+         tracks.inside=10, tracks.outside=0);
> #
> #        Reduce text size by modifying plot parameters
> #        ================================================
> params <- RCircos.Get.Plot.Parameters();
> params$text.size <- 0.3;
> RCircos.Reset.Plot.Parameters(params);
> #
> #        Plot chromosome ideogram
> #        ================================================
> RCircos.Set.Plot.Area()
> RCircos.Chromosome.Ideogram.Plot()
> #
> #          Reset colors for some gene names
> #          ================================================
> data(RCircos.Gene.Label.Data);
> gene.data <- RCircos.Gene.Label.Data;
> gene.colors <- rep("black", nrow(gene.data))
> gene.colors[which(gene.data$Gene=="TP53")] <- "red";
> gene.colors[which(gene.data$Gene=="BRCA2")] <- "red";
> gene.colors[which(gene.data$Gene=="RB1")] <- "red";
> gene.colors[which(gene.data$Gene=="JAK1")] <- "red";
> gene.colors[which(gene.data$Gene=="JAK2")] <- "red";
> gene.colors[which(gene.data$Chromosome=="chr4")] <- "blue";
> gene.data["PlotColor"] <- gene.colors;
> RCircos.Gene.Connector.Plot(gene.data, track.num=6,
+                         side="in");
> RCircos.Gene.Name.Plot(gene.data, name.col=4,
+                 track.num=2, side="in");
> #
> #        Heatmap plot with default color map
> #        ================================================
> data(RCircos.Heatmap.Data);
> RCircos.Heatmap.Plot(RCircos.Heatmap.Data, data.col=6,
+                 track.num=5, side="in");
> #
> #        Reset colors for histogram plot
> #        ================================================
> data(RCircos.Histogram.Data);
> hist.data <- RCircos.Histogram.Data;
> hist.colors <- rep("blue", nrow(hist.data))
> rows <- which(hist.data$Data>0.4)
> hist.colors[rows] <- "red";
> hist.data["PlotColor"] <- hist.colors;
> RCircos.Histogram.Plot(hist.data, data.col=4,
+         track.num=6, side="in");
> #
> #        Reset colors for line plot
> #        ================================================
> data(RCircos.Line.Data);
> line.data <- RCircos.Line.Data;
> line.colors <- rep("blue", nrow(line.data))
> rows <- which(abs(line.data$seg.mean)>=1.5);
> line.colors[rows] <- "red";
> line.data["PlotColor"] <- line.colors;
> RCircos.Line.Plot(line.data, data.col=5, track.num=7,
+                 side="in");
> #
> #        Reset colors for scatter plot. Note: argument
> #        by.fold must set to 0.
> #        ================================================
> data(RCircos.Scatter.Data);
> scatter.data <- RCircos.Scatter.Data;
> scatter.colors <- rep("yellow", nrow(scatter.data));
> scatter.colors[which(scatter.data$seg.mean>=1)] <- "red";
> scatter.colors[which(scatter.data$seg.mean<=-1)] <- "blue";
> scatter.data["PlotColor"] <- scatter.colors;
> RCircos.Scatter.Plot(scatter.data, data.col=5, track.num=8,
+         side="in", by.fold=0);
> #
> #        Reset colors for tile plot
> #        ================================================
> data(RCircos.Tile.Data);
> tile.data <- RCircos.Tile.Data;
> tile.data["PlotColor"] <- rep("cyan", nrow(tile.data));
> RCircos.Tile.Plot(tile.data, track.num=9, side="in");
> #
> #        Reset colors for link lines. Note: argument
> #        by.chromosome must be set to FALSE
> #        ================================================
> data(RCircos.Link.Data);
> link.data <- RCircos.Link.Data;
> link.colors <- rainbow(nrow(link.data));
> link.data["PlotColor"] <- link.colors;
> RCircos.Link.Plot(link.data, track.num=11, by.chromosome=FALSE);
> #
> #        Reset colors for ribbons. Note: argument
> #        by.chromosome must be set to FALSE
> #        ================================================
> data(RCircos.Ribbon.Data);
> ribbon.colors <- c(rgb(1, 0, 0, alpha=0.5),
+                     rgb(0, 1, 0, alpha=0.5),
+                    rgb(0, 0, 1, alpha=0.5),
+                     rgb(1, 1, 0, alpha=0.5));
> ribbon.data <- RCircos.Ribbon.Data;
> ribbon.data["PlotColor"] <- ribbon.colors;
> RCircos.Ribbon.Plot(RCircos.Ribbon.Data, track.num=11,
+         by.chromosome=FALSE, twist=FALSE);
> #
> title("RCircos Plot Color Demo")
```



3 Working with RCircos.Cytoband Object

3.1 The RCircos.Cytoband Object

Chromosome ideogram is the core track of RCircos plot. RCircos uses RCircos.Cytoband Object to hold the ideogram information. The RCircos.Cytoband Object is a data frame containing following columns:

Chromosome chromosome names

ChromStart the start base pair from the begining of the chromosome

ChromEnd the end base pair from the begining of the chromosome

Band cytoband name

Stain Giemsa stain color

BandColor colors to be plotted for this band

ChrColor highlight color for the chromosome

Length length of the band in base pairs

Unit length of the band in chromosome unit

Location the last position of the band on the circle

The first five columns are from original chromosome ideogram data and should never be changed. Other five columns are assigned by RCircos after ran RCircos.Set.Core.Components()function. Users may modify the columns of BandColor, chrColor, and Location to make special plots.

3.2 Working with Genomes of other species

Currently, RCircos only provides three build-in chromosome ideogram data sets. For other species, RCircos will work well if the chromosome ideogram data could be supplied in same data format. Unfortunately, many species do not have chromosome band information. Next update of RCircos will be configured out to work with chromosome ideogram data without band information. Currently, users can follow procedures below to make RCircos.Cytoband Object for plotting.

1). Manually edit the ideogram data or load it into R then construct a data frame having five columns as the build-in ideogram. The cytoband name could be any names and stain could be any one used in the build-in ideogram.

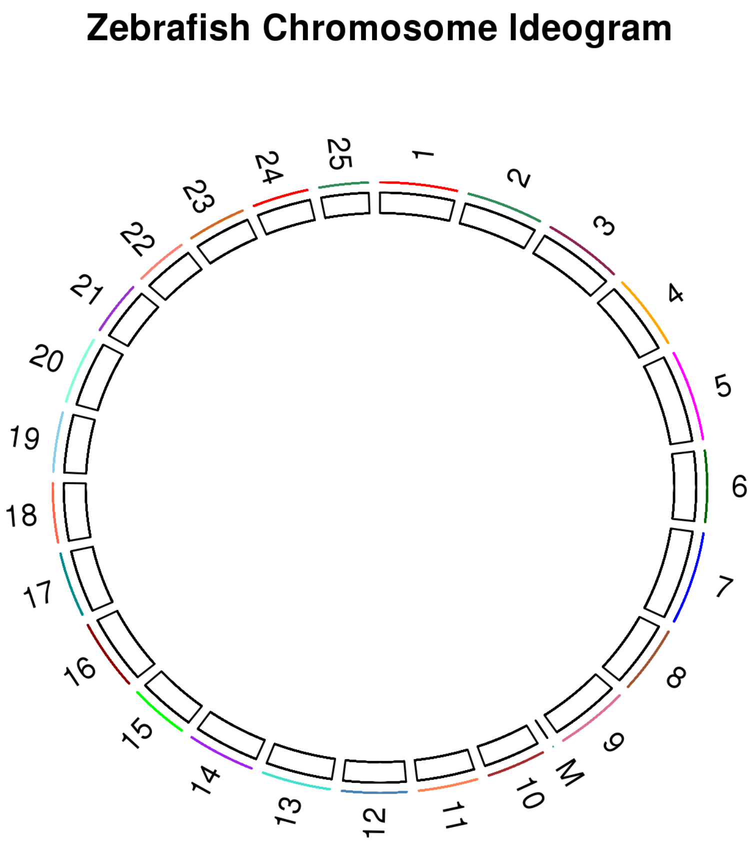
2). Bypass the RCircos.Set.Core.Components() function with following code:

```
> RCircos.Initialize.Parameters(tracks.inside=5,tracks.outside=0);
> RCircos.Set.Cytoband.data(chr.info);
> RCircos.Set.Base.Plot.Positions();
```

3). Make RCircos plot as usual.

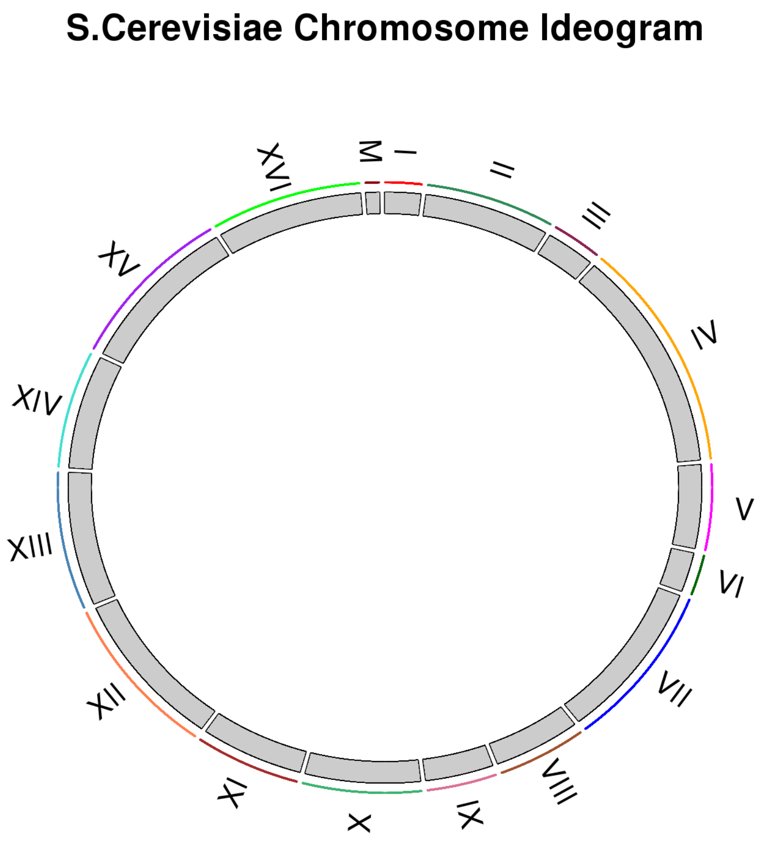
Run code below will get RCircos plot image with Zebrafish chromosome ideogram

```
> #
> #        Load RCircos library and chromosome ideogram data
> #        ===============================================
> library(RCircos);
> ideo.table <- read.table("Zebrafish.ideogram.txt",
+         header=F, sep="\t", quote="");
> ideogram <- ideo.table[grep("chr", ideo.table[,1]),];
> colnames(ideogram) <- c("Chromosome", "ChromStart",
+         "ChromEnd", "Band", "Stain");
> ideogram$Band <- paste0(ideogram$Chromosome, "Band");
> #
> #        Bypass the RCircos.Set.Core.Components()
> #        ========================================
> RCircos.Initialize.Parameters(tracks.inside=5,
+                         tracks.outside=0);
> RCircos.Set.Cytoband.data(ideogram);
> RCircos.Set.Base.Plot.Positions();
> #
> #        Plot chromosome ideogram
> #        ==============================================
> RCircos.Set.Plot.Area();
> RCircos.Chromosome.Ideogram.Plot();
> title("Zebrafish Chromosome Ideogram");
```



By combining both of bypassing the RCircos.Set.Core.Components() function and modifying relevant parameter(s), RCircos can plot even very small genomes without band inforamtion.

```
> #
> #        Load RCircos library and chromosome ideogram data
> #        ================================================
> library(RCircos);
> ideo.table <- read.table("S.Cerevisuae.Genome.txt",
+          header=F, sep="\t", quote="");
> ideogram <- ideo.table[,c(2, 3, 4, 7)];
> colnames(ideogram) <- c("Chromosome", "ChromStart",
+          "ChromEnd", "Band");
> ideogram["Stain"] <- rep("gpos25", nrow(ideogram));
> #
> #        Sort the chromosome names
> #        ===============================================
> #
> row.index <- c(1, 17, 5, 10, 2, 4, 9, 6, 11, 15, 3,
+                 7, 12, 14, 16, 8, 13);
> ideogram <- ideogram[order(row.index),];
> head(ideogram);
> #
> #        Bypass the RCircos.Set.Core.Components()
> #        ================================================
> #
> RCircos.Initialize.Parameters(tracks.inside=5,
+                         tracks.outside=0);
> RCircos.Set.Cytoband.data(ideogram);
> RCircos.Set.Base.Plot.Positions();
> #
> #        Reduce chromosome unit size from 3000 to 30
> #        since the genome is very small
> #        ================================================
> #
> params <- RCircos.Get.Plot.Parameters();
> params$base.per.unit <- 30;
> RCircos.Reset.Plot.Parameters(params);
> #
> #        Plot chromosome ideogram
> #        ================================================
> RCircos.Set.Plot.Area();
> RCircos.Chromosome.Ideogram.Plot();
> title("S.Cerevisiae Chromosome Ideogram");
```

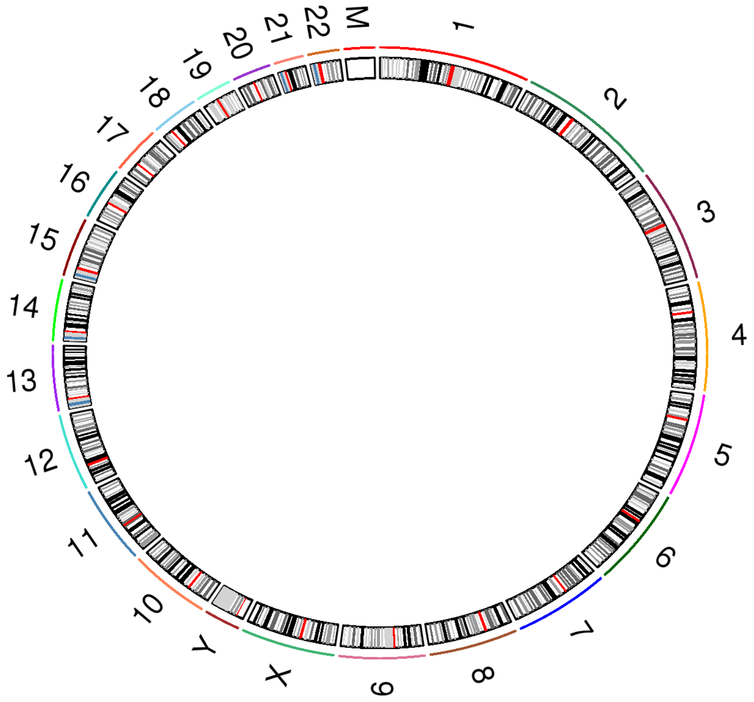


3.3 Adding chromosome(s) to ideogram

In case another chromosome needs to be added to chromosome ideogram such as human mitochondria chromosome, user can append relevant information to the chromosome ideogram data either in file or the data frame in R session. Note: since the length of mitochondria chromosome is very small, it must be scaled in both ideogram data and plot data to make it readable on the image.

Following image is generated with UCSC.HG19.Human.CytoBandIdeogram combined with mitochondria chromosomes (scaled from 16,569 base pairs to 49500000 base pairs).

```
> #
> #        Load RCircos library and chromosome ideogram data
> #        ================================================
> library(RCircos);
> data(UCSC.HG19.Human.CytoBandIdeogram);
> cyto.info <- UCSC.HG19.Human.CytoBandIdeogram;
> #
> #        Read in mitochondria chromosome information
> #        ================================================
> chr.M.file <- "chrM.ideo.data.txt";
> chr.M.info <- read.table(chr.M.file, header=T, sep="\t", quote="")
> colnames(chr.M.info) <- c("Chromosome", "ChromStart",
+                 "ChromEnd", "Band", "Stain");
> #
> #        Append mitochondria info to hg19 ideogram
> #        ================================================
> cyto.info <- rbind(cyto.info, chr.M.info);
> #
> #        Bypass the RCircos.Set.Core.Components()
> #        ================================================
> RCircos.Initialize.Parameters(tracks.inside=5,tracks.outside=0);
> RCircos.Set.Cytoband.data(cyto.info);
> RCircos.Set.Base.Plot.Positions();
> #
> #        Plot chromosome ideogram
> #        ================================================
> RCircos.Set.Plot.Area();
> RCircos.Chromosome.Ideogram.Plot();
> title("Human Chromosome Ideogram with ChrM");
```



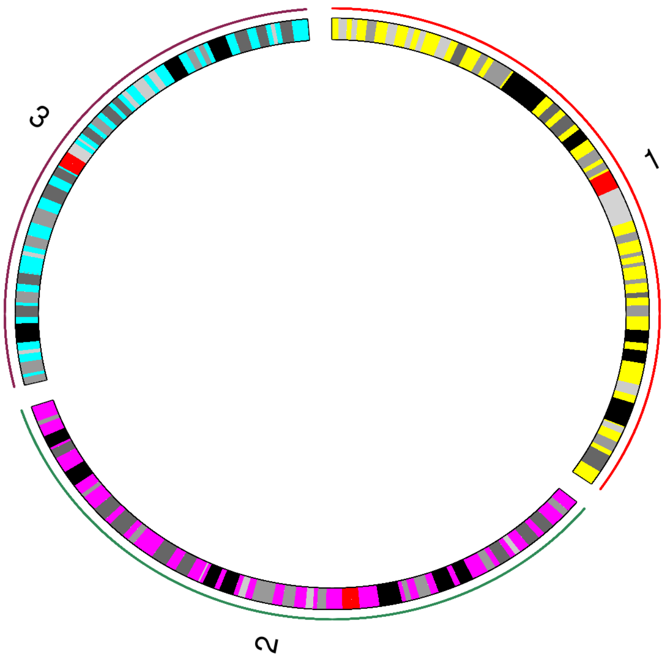
Human Chromosome Ideogram with ChrM

3.4 Change Chromosome Band Colors

RCircos holds chromosome band colors in RCircos.Cytoband object and uses gray scale colors to represent Giemsa staining intensities. The band colors could be changed in order to remark data along a specific chromosome band.

```
> #
> #        Load RCircos library and chromosome ideogram data
> #        ================================================
> library(RCircos);
> data(UCSC.HG19.Human.CytoBandIdeogram);
> human.cyto <- UCSC.HG19.Human.CytoBandIdeogram;
> #
> #        Initialize RCircos core components and keep
> #        chromosome 1, 2, and 3 only
> #        ===============================================
> chr.exclude <- paste0("chr", c(4:22, "X", "Y"))
> RCircos.Set.Core.Components(human.cyto,chr.exclude=NULL,
+                 tracks.inside=5, tracks.outside=0);
> #
> #        Reset colors for white bands
> #        ===============================================
> cyto.band <- RCircos.Get.Plot.Ideogram()
> #
> chr1.white <- cyto.band$Chromosome=="chr1" &
+         cyto.band$BandColor=="white";
> cyto.band$BandColor[chr1.white==T] <- "yellow";
> #
> chr2.white <- cyto.band$Chromosome=="chr2" &
+         cyto.band$BandColor=="white";
> cyto.band$BandColor[chr2.white==T] <- "magenta";
> #
> chr3.white <- cyto.band$Chromosome=="chr3" &
+         cyto.band$BandColor=="white";
> cyto.band$BandColor[chr3.white==T] <- "cyan";
> #
> RCircos.Reset.Plot.Ideogram(cyto.band)
> #
> #        Plot chromosome ideogram
> #        ===============================================
> RCircos.Set.Plot.Area()
> RCircos.Chromosome.Ideogram.Plot()
> #
> title("RCircos Demo: Change Band Color")
```

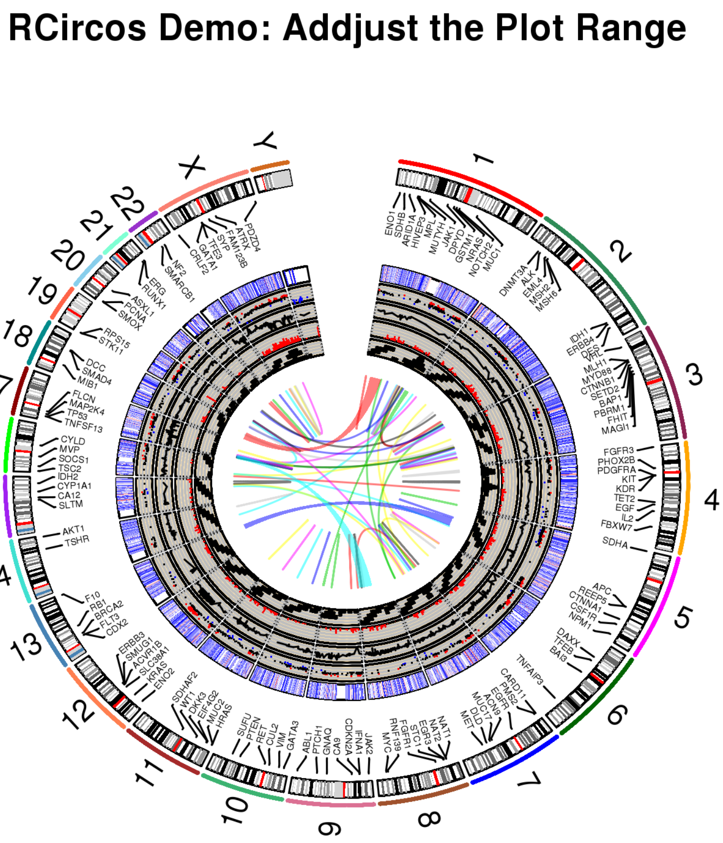
RCircos Demo: Change Band Color



3.5 Modifying Plot Ranges

By default, RCircos fills data tracks to all area. In case some small empty area are needed for adding other lables later, items in the RCircos.Cytoband object could be modified so that data tracks will be plotted on a portion of the plot area.

```
> #
> #        Load RCircos library and chromosome ideogram data
> #        ===================================================
> library(RCircos);
> data(UCSC.HG19.Human.CytoBandIdeogram);
> cyto.info <- UCSC.HG19.Human.CytoBandIdeogram;
> #
> #        Initialize RCircos core components
> #        ==================================================
> RCircos.Set.Core.Components(cyto.info, chr.exclude=NULL,
+                 tracks.inside=10, tracks.outside=0);
> #
> #        Get the three RCircos core components
> #        ================================================
> plot.ideo <- RCircos.Get.Plot.Ideogram();
> plot.pos  <- RCircos.Get.Plot.Positions();
> plot.par  <- RCircos.Get.Plot.Parameters();
> #
> #        Reduce the text size since there will be only
> #        90% rooms
> #        ================================================
> #
> plot.par$text.size <- 0.3;
> RCircos.Reset.Plot.Parameters(plot.par);
> #
> #        Calculate empty space size and location
> #        ================================================
> #
> empty.space <- 0.05;
> scale.factor <- 1 - empty.space;
> total.units <- dim(plot.pos)[1];
> space.total <- round(total.units*empty.space, digits=0);
> space.half <- round(space.total/2, digits=0);
> #
> #        Reset chromosome ideogram object
> #        ================================================
> #
> new.ideogram <- plot.ideo;
> new.ideogram$Unit <- round(new.ideogram$Unit*scale.factor, digits=0);
> new.ideogram$Location <- round(new.ideogram$Location*scale.factor, digits=0);
> new.ideogram$Location <- new.ideogram$Location + space.half;
> RCircos.Reset.Plot.Ideogram(new.ideogram);
> #
> #        Plot chromosome ideogram
> #        ================================================
> RCircos.Set.Plot.Area();
> title("RCircos Demo: Addjust the Plot Range");
> RCircos.Chromosome.Ideogram.Plot();
> #
> #        Gene name and connector plot
> #        ================================================
> data(RCircos.Gene.Label.Data);
> RCircos.Gene.Connector.Plot(RCircos.Gene.Label.Data,
+                         track.num=1, side="in");
> RCircos.Gene.Name.Plot(RCircos.Gene.Label.Data, name.col=4,
+                         track.num=2, side="in");
> #
> #        Heatmap plot
> #        ================================================
> data(RCircos.Heatmap.Data);
> RCircos.Heatmap.Plot(RCircos.Heatmap.Data, data.col=6,
+                         track.num=5, side="in");
> #
> #        Scatter plot
> #        ================================================
> data(RCircos.Scatter.Data);
> RCircos.Scatter.Plot(RCircos.Scatter.Data, data.col=5,
+                         track.num=6, side="in", by.fold=1);
> #
> #        Line plot
> #        ================================================
> data(RCircos.Line.Data);
> RCircos.Line.Plot(RCircos.Line.Data, data.col=5,
+                         track.num=7, side="in");
> #
> #        Histogram plot
> #        ================================================
> data(RCircos.Histogram.Data);
> RCircos.Histogram.Plot(RCircos.Histogram.Data, data.col=4,
+                         track.num=8, side="in");
> #
> #        Tile plot
> #        ================================================
> data(RCircos.Tile.Data);
> RCircos.Tile.Plot(tile.data=RCircos.Tile.Data, track.num=9, side="in");
> #
> #        Link line plot
> #        ================================================
> data(RCircos.Link.Data);
> RCircos.Link.Plot(RCircos.Link.Data, track.num=11,
+                         by.chromosome=FALSE);
> #
> #        Ribbon link plot
> #        ================================================
> data(RCircos.Ribbon.Data);
> RCircos.Ribbon.Plot(RCircos.Ribbon.Data, track.num=11,
+                 by.chromosome=FALSE, twist=FALSE);
```



RCircos Demo: Addjust the Plot Range

4 Working with RCircos.Base.Position Object

4.1 The RCircos.Base.Position Object

RCircos.Base.Position is a data frame. The first and second columns are for x and y coordinates of points for a circle line, the third column is degrees that a character should be rotated for text plot. This object are usually used for calculation of location of data points to be plotted.

```
> #
> #       Load RCircos library and chromosome ideogram data
> #       ==================================================
> library(RCircos);
> data(UCSC.HG19.Human.CytoBandIdeogram);
> cyto.info <- UCSC.HG19.Human.CytoBandIdeogram;
> #
> #        Initialize RCircos core components
> #        ================================================
> RCircos.Set.Core.Components(cyto.info,chr.exclude=NULL,
+         tracks.inside=5, tracks.outside=0);

RCircos.Core.Components initialized.
Type ?RCircos.Reset.Plot.Parameters to see how to modify the core components.

> #
> #        Get RCircos plot position object
> #        ================================================
> positions <- RCircos.Get.Plot.Positions();
> head(positions);

         cor.x cor.y   degree
1 0.000000e+00     1 90.00000
2 5.691786e-06     1 89.99967
3 1.138357e-05     1 89.99935
4 1.707536e-05     1 89.99902
5 2.276714e-05     1 89.99870
6 2.845893e-05     1 89.99837
```

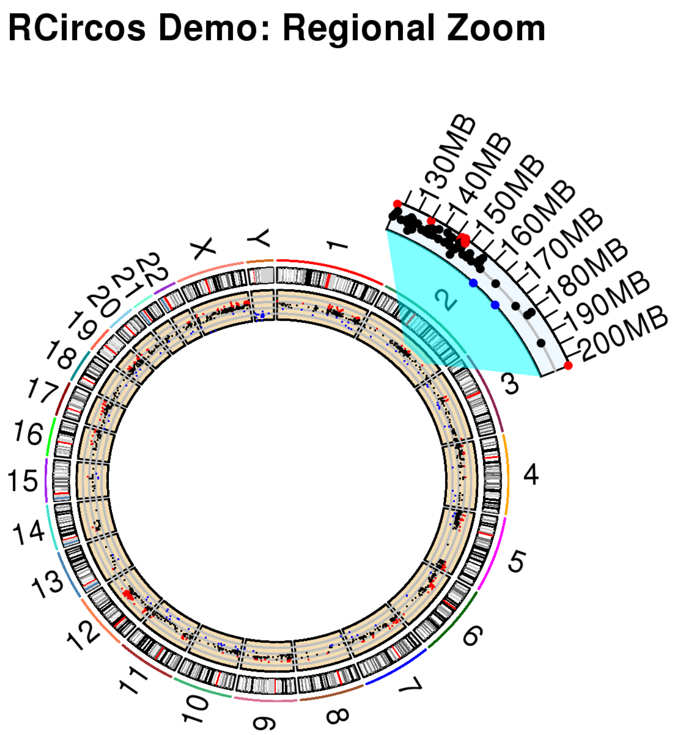
4.2 Add More Decoration Items

By scaling and indexing of the RCircos.Base.Position object, one can easily to get plot positions for any point on the plot image and add user's own graphic items.

Following code shows how to add a zoom-in regional track by working with the RCircos.PlotPar and RCircos.Base.Positions objects.

```
> #
> #        Load RCircos library and chromosome ideogram data
> #        =================================================
> library(RCircos);
> data(UCSC.HG19.Human.CytoBandIdeogram)
> data(RCircos.Scatter.Data);
> #
> #        Initialize RCircos core components
> #        ================================================
> human.cyto <- UCSC.HG19.Human.CytoBandIdeogram
> RCircos.Set.Core.Components(human.cyto, chr.exclude=NULL,
+         tracks.inside=5, tracks.outside=5);
> #
> #        Plot chromosome ideogram
> #        ================================================
> RCircos.Set.Plot.Area();
> RCircos.Chromosome.Ideogram.Plot();
> #
> #        Get the three RCircos core components
> #        ==================================================
> cytoband <- RCircos.Get.Plot.Ideogram();
> positions <- RCircos.Get.Plot.Positions();
> params <- RCircos.Get.Plot.Parameters();
> #
> #        Reset track height
> #        ==================================================
> params$track.height <- params$track.height*2;
> RCircos.Reset.Plot.Parameters(params);
> #
> #        Scatter plot on a track with doubled height
> #        ================================================
> RCircos.Scatter.Plot(RCircos.Scatter.Data, data.col=5,
+                 track.num=1, side="in", by.fold=1);
> #
> #        Set zoom area (1/8 of the circle circumference)
> #        ==================================================
> area.len <- round(dim(positions)[1]/8, digits=0)
> area.start <- round(area.len/2, digits=0);
> area.end <- area.start + area.len;
> #
> in.pos <- params$track.out.start
> middle <- in.pos + params$track.height/2;
> out.pos <- in.pos + params$track.height;
> #
> #        Select the chromosome 2 region to zoom in
> #        ==================================================
> chr2.rows <- which(RCircos.Scatter.Data[,1]=="2");
> chr2.data <- RCircos.Scatter.Data[chr2.rows,];
> cn2.rows <- which(chr2.data[,5]>=1);
> #
> seg.start <- 122700000;
> seg.end   <- 203020000;
> #
> zoom.start <- min(which(chr2.data[,2] >= seg.start));
> zoom.end   <- max(which(chr2.data[,3] <= seg.end));
> zoom.data <- chr2.data[zoom.start:zoom.end,];
> #
> #        Index of the plot position after zoom in
> #        ==================================================
> total.basepairs <- seg.end - seg.start +  1;
> bp.per.point <- round(total.basepairs/area.len, digits=0);
> #
> zoom.pos  <- round((zoom.data[,3]-zoom.data[,2])/2,
+                 digits=2) + zoom.data[,2];
> zoom.pos <- round((zoom.pos-seg.start)/ bp.per.point,
+                 digits=0) + area.start;
> #
> #        Outline the zoom area
> #        ==================================================
> polygon.x <- c(positions[area.start:area.end, 1]*out.pos,
+                 positions[area.end:area.start,1]*in.pos);
> polygon.y<- c(positions[area.start:area.end,2]*out.pos,
+                 positions[area.end:area.start,2]*in.pos);
> polygon(polygon.x, polygon.y, col="aliceblue");
> lines(positions[area.start:area.end,1]*middle,
+         positions[area.start:area.end,2]*middle,
+         col="gray");
> #
> #        Link the plot area and zoom area
> #        ==================================================
> data.pos <- params$track.in.start - params$track.heigh/3;
> point.one <- RCircos.Data.Point("chr2", seg.start);
> point.two <- RCircos.Data.Point("chr2", seg.end);
> polygon(c(positions[area.end:area.start,1]*in.pos,
+                 positions[point.one:point.two, 1]*data.pos),
+         c(positions[area.end:area.start,2]*in.pos,
+                 positions[point.one:point.two, 2]*data.pos),
+         col=rgb(0, 1, 1, 0.5), border=NA);
> #
> #        Plot the data point on zoom area
> #        ==================================================
> data.ceiling <- 1.5;
> data.col <- 5;
> subtrack.height <- out.pos - middle;
> #
> for(a.point in 1:nrow(zoom.data))
+ {
+         #
+         #        Avoid data overflow
+         #        ==================================
+         the.data <- zoom.data[a.point, data.col];
+         if(the.data > data.ceiling) {
+                 the.value<- data.ceiling;
+         } else if(the.data <(-1*data.ceiling)) {
+                 the.value <- data.ceiling*-1;
+         } else { the.value <- the.data;  }
+         #
+         #        Assign color based on lo2 values
+         #        =================================
+         by.fold <- 1;
+         if(the.value>=by.fold) {
+                 color <- "red";
+         } else if (the.value<=-by.fold) {
+                 color <- "blue";
+         } else {  color <- "black"; }
+         #
+         #        Plot the point
+         #        =========================================
+         adjust <- the.value/data.ceiling*subtrack.height;
+         height <- middle  +  adjust;
+         points(positions[zoom.pos[a.point], 1]*height,
+                 positions[zoom.pos[a.point], 2]*height,
+                 col=color, pch=19, cex=0.5);
+ }
> #
> #        Add ticks on zoom area (a long tick every 40MB bps)
> #        by indexing RCircos.Base.Position object
> #        ==================================================
> tick.length <- 10000000;
> long.width <-round(tick.length/bp.per.point, digits=0);
> short.width <- round(long.width/2, digits=0);
> total.long.ticks <- ceiling(area.len/long.width);
> total.short.ticks <- ceiling(area.len/short.width);
> #
> #        Tick and lable poisitions
> #        ==================================================
> long.top <- out.pos + (middle-in.pos);
> short.top <- long.top - (middle-in.pos)/2;
> #
> zoom.area <- positions[area.start:area.end,];
> tick.start <- zoom.area[,1:2]*out.pos;
> long.tick.end <- zoom.area[,1:2]*long.top;
> short.tick.end <- zoom.area[,1:2]*short.top;
> #
> lab.pos <- zoom.area*(out.pos + params$track.height*2);
> lab.pos[,3] <- zoom.area[,3];
> #
> #        Plot ticks and labels
> #        ===========================================================
> num.long <- ceiling(seg.start/tick.length);
> extra.points <- (num.long*tick.length - seg.start)/bp.per.point;
> first.long <- round(extra.points, digits=0);
> for(a.tick in 1:total.long.ticks)
+ {
```

```
+    tick.pos <- first.long + (a.tick-1)*long.width;
+    if(tick.pos>area.len) {break; }
+
+    lines(c(tick.start[tick.pos,1], long.tick.end[tick.pos,1]),
+        c(tick.start[tick.pos,2], long.tick.end[tick.pos,2]),
+        col="black");
+    lab.text <- paste0((num.long+a.tick-1)*10, "MB");
+    text(lab.pos[tick.pos,1] ,  lab.pos[tick.pos,2],
+        lab.text, srt=lab.pos[tick.pos, 3]);
+ }
> num.short <- ceiling(seg.start/(tick.length/2));
> extra.points <- (num.short*(tick.length/2) - seg.start)/bp.per.point;
> first.short <- round(extra.points, digits=0);
> for(a.tick in 1:total.short.ticks)
+ {
+    tick.pos <- first.short + (a.tick-1)*long.width;
+    lines(c(tick.start[tick.pos,1], long.tick.end[tick.pos,1]),
+        c(tick.start[tick.pos,2], long.tick.end[tick.pos,2]),
+        col="black");
+ }
> #
> title("RCircos Demo: Regional Zoom")
```



5 SessionInfo

```
> sessionInfo()

R version 3.0.1 (2013-05-16)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
[1] C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] RCircos_1.1.2
```

```
loaded via a namespace (and not attached):
[1] tools_3.0.1
```

References

[1] Honge Zhang, Paul Meltzer, and Sean Davis. RCircos: An R package for Circos 2D track plots. *BMC Bioinformatics*, 2013.

[2] Krzywinski, Martin I and Schein, Jacqueline E and Birol, Inanc and Connors, Joseph and Gascoyne, Randy and Horsman, Doug and Jones, Steven J and Marra, Marco A. Circos: An information aesthetic for comparative genomics. *Genome Research*, 2009.