IdeoViz
Plot data along chromosome ideograms

Shraddha Pai (shraddha.pai@utoronto.ca), Jingliang Ren

8 December, 2017

Plotting discrete or continuous dataseries in the context of chromosomal location has several useful applications in genomic analysis. Examples of possible metrics include RNA expression levels, densities of epigenetic marks or genomic variation, while applications could range from the analysis of a single variable in a single context, to multiple measurements in several biological contexts (e.g. age/sex/tissue/disease context). Visualization of metrics superimposed on the chromosomal ideogram could provide varied insights into the metric of interest:

1. It could identify distinctive spatial distribution that could further hypotheses about the functional role of the metric (e.g. telocentric or pericentromeric enrichment)
2. It could highlight distribution differences between different groups of samples, suggesting different regulatory mechanisms; in extreme cases, visualization may identify large genomic foci of differences
3. It could confirm that a quantitative difference measured between groups of interest is consistent throughout the genome (i.e. that there are no foci, and that the change is global).

This package provides a method to plot one or several dataseries against the chromosomal ideogram. It provides some simple options (vertical/horizontal orientation, display in bars or linegraphs). Data are expected to be binned; IdeoViz provides a function for user-specified bin widths. Ideograms for the genome of choice can also be automatically downloaded from UCSC using the getIdeo() function.

1 Setup

```
> require(IdeoViz)
> require(RColorBrewer) ### nice colours
> data(binned_multiSeries)
```

2 Example 1: Plotting several trendlines along one ideogram

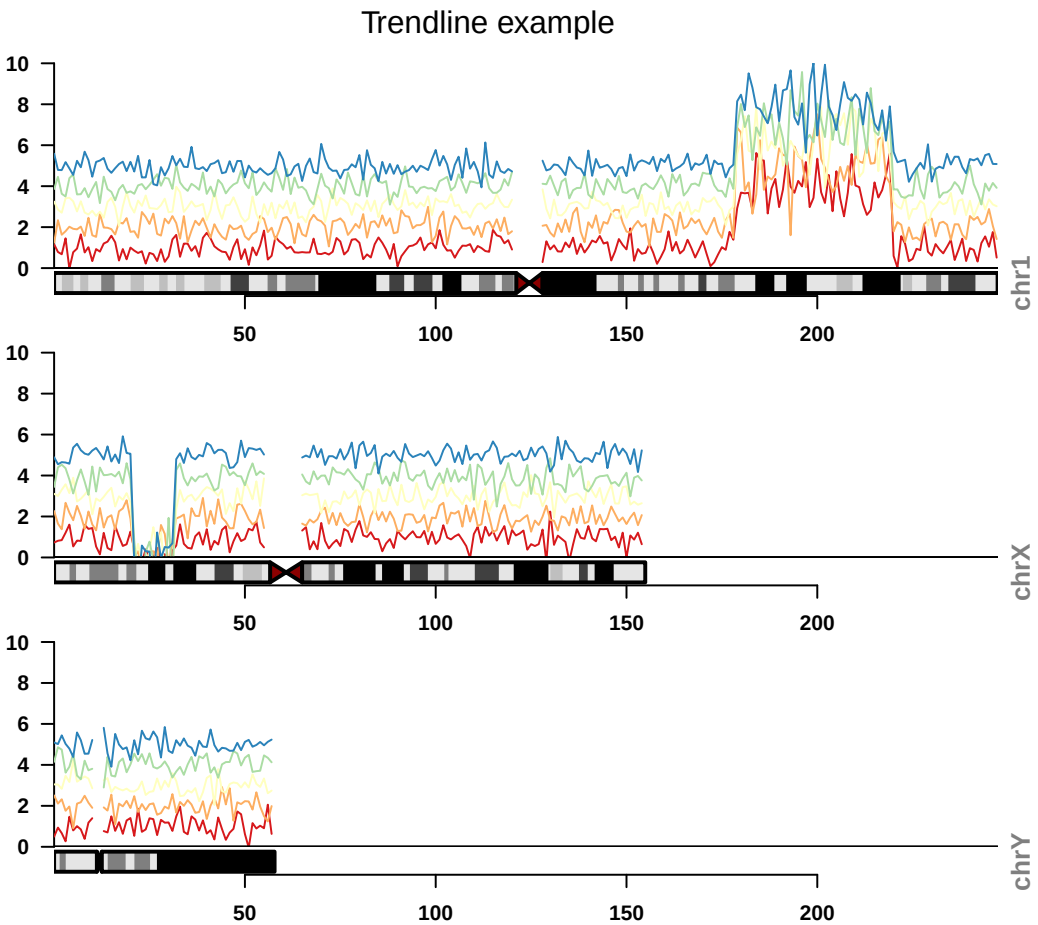
The ideogram table containing cytogenetic band information is used to render chromosomes. This table corresponds directly to the *cytoBandIdeo* table from the UCSC genome browser. There are two ways to supply an ideogram table to *plotOnIdeo()*:

1. First, it can be automatically downloaded from UCSC for your genome of choice, using the *getIdeo()* function.
2. Alternately, a pre-downloaded *cytoBandIdeo* table can be provided to downstream functions such as *plotOnIdeo()*. In this case, the table must be provided as a data.frame object with a header row and the column order matching that of the *cytoBandIdeo()* table at UCSC.

```
> ideo <- getIdeo("hg18")
> head(ideo)
```

```
  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
```

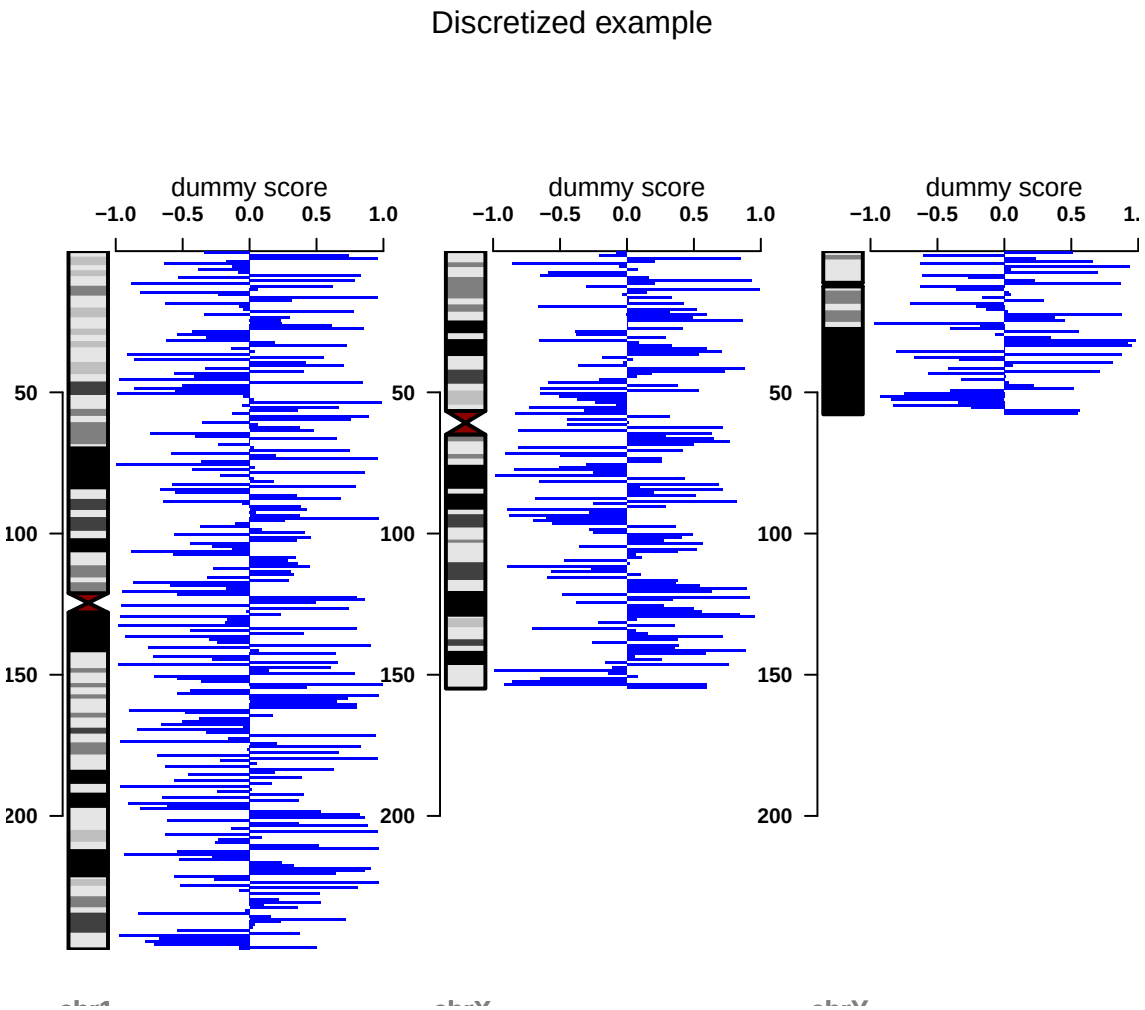
```
> plotOnIdeo(chrom=seqlevels(binned_multiSeries), # which chrom to plot?
+            ideoTable=ideo, # ideogram name
+            values_GR=binned_multiSeries, # data goes here
+            value_cols=colnames(mcols(binned_multiSeries)), # col to plot
+            col=brewer.pal(n=5, 'Spectral'), # colours
+            val_range=c(0,10), # set y-axis range
+            ylab="array intensities",
+            plot_title="Trendline example")
```



Example 2: Plotting a single series in bar format

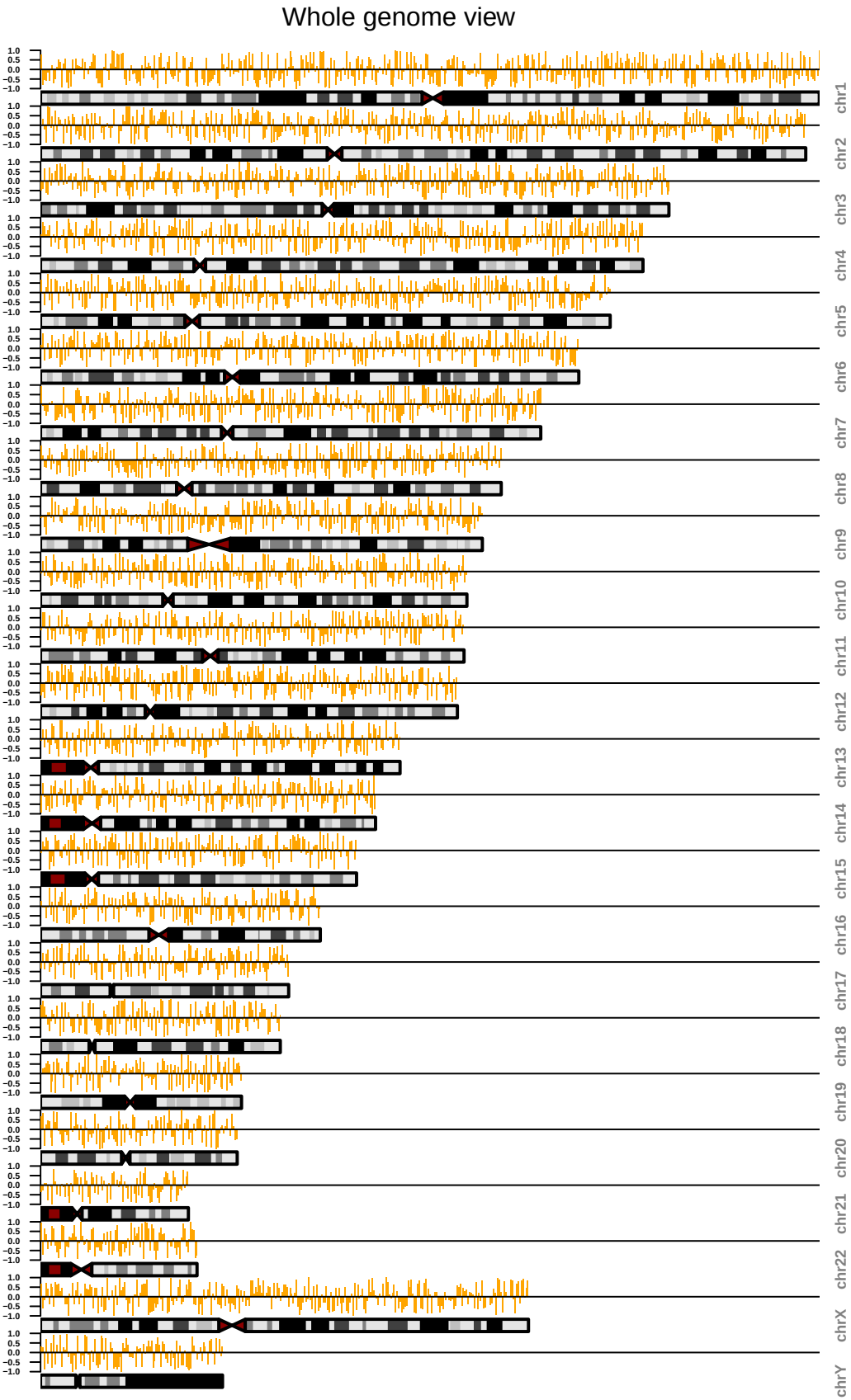
For this example, we specify a local file to obtain the chromosome ideograms, rather than having IdeoViz download it from UCSC.

```
> data(binned_singleSeries)
> data(hg18_ideo) # cytoBandIdeo table downloaded previously and stored as a data.frame.
> plotOnIdeo(chrom=seqlevels(binned_singleSeries),
+            ideo=hg18_ideo,
+            values_GR=binned_singleSeries,
+            value_cols=colnames(mcols(binned_singleSeries)),
+            plotType='rect', # plot as bars
+            col='blue', vertical=T,
+            val_range=c(-1,1), ylab="dummy score",
+            plot_title="Discretized example")

Plot chromosome done
Plot chromosome done
Plot chromosome done
```



Example 3: Plotting a single series in bar format along entire genome

```
> data(binned_fullGenome)
> plotOnIdeo(chrom=seqlevels(binned_fullGenome),
+           ideo=ideo,
+           values_GR=binned_fullGenome,
+           value_cols=colnames(mcols(binned_fullGenome)),
+           plotType='rect',
+           col='orange', addScale=F, # hide scale to remove visual clutter
+           plot_title="Whole genome view",
+           val_range=c(-1,1),cex.axis=0.5,chromName_cex=0.6)
```



3 Example 4: Binning data using IdeoViz functions

In this example, we do everything in IdeoViz: download the ideogram from UCSC, bin the data, and finally, plot along chromosomes. For the example, we use histone H3K9me3 peak intensities mapped in the human lymphoblastoid cell line

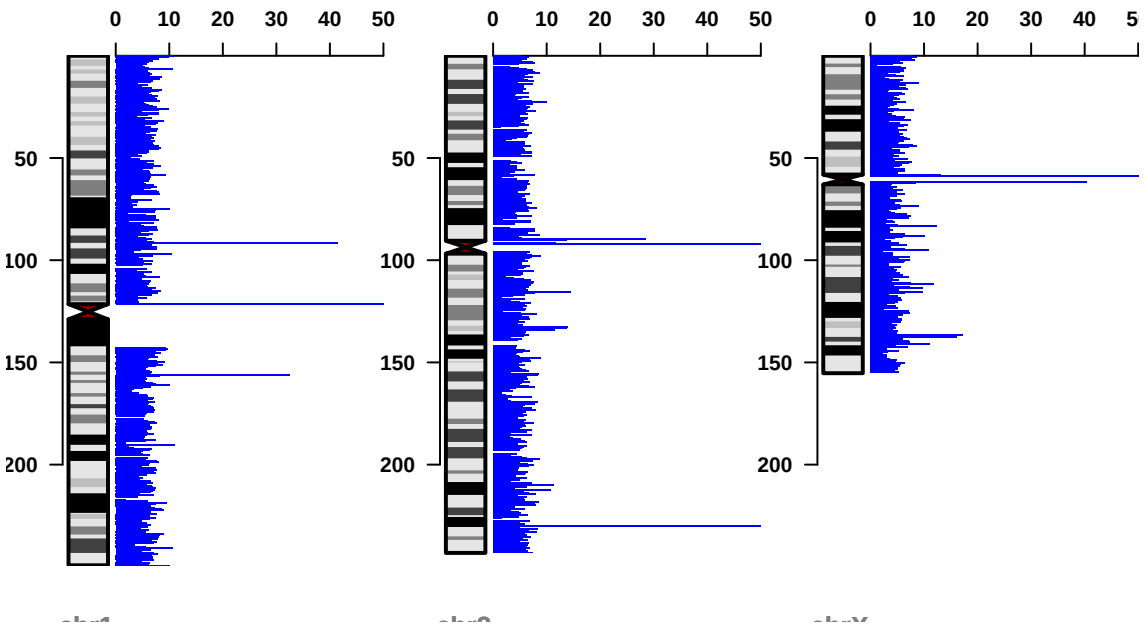
GM12878 (GEO accession GSM733664, only 3 chromosomes shown for simplicity). Here, average peak signal is plotted in 500Kb bins along the chromosome. The ideogram plots show high signal in pericentromeric and telomeric regions, consistent with the association of this histone mark with heterochromatin.
Reference: ENCODE Project Consortium, Bernstein BE, Birney E, Dunham I, Green ED, Gunter C, Snyder M. An integrated encyclopedia of DNA elements in the human genome. *Nature.*(2012): **489** (7414):57-74.

```
> ideo_hg19 <- getIdeo("hg19")
> chroms <- c("chr1","chr2","chrX")
> data(GSM733664_broadPeaks)
> head(GSM733664_broadPeaks)
```

```
  chrom chromStart chromEnd name score strand signalValue pValue qValue
1  chr1      10141    10374    .   993      .   10.796883   10.3     -1
2  chr1     567457   567702    .  1000      .   16.590333  100.0     -1
3  chr1     569826   570047    .  1000      .   15.757614  100.0     -1
4  chr1     723167   727602    .   808      .    8.389733   14.2     -1
5  chr1     816959   817136    .   793      .    8.188648    1.7     -1
6  chr1     821181   821421    .   753      .    7.660859    3.4     -1
```

```
> chrom_bins <- getBins(chroms, ideo_hg19,stepSize=5*100*1000)
> avg_peak <- avgByBin(data.frame(value=GSM733664_broadPeaks[,7]),
+   GSM733664_broadPeaks[,1:3], chrom_bins)
> plotOnIdeo(chrom=seqlevels(chrom_bins),
+           ideoTable=ideo_hg19,
+           values_GR=avg_peak, value_cols='value',
+           val_range=c(0,50),
+           plotType='rect',
+           col='blue', vertical=T
+ )
```

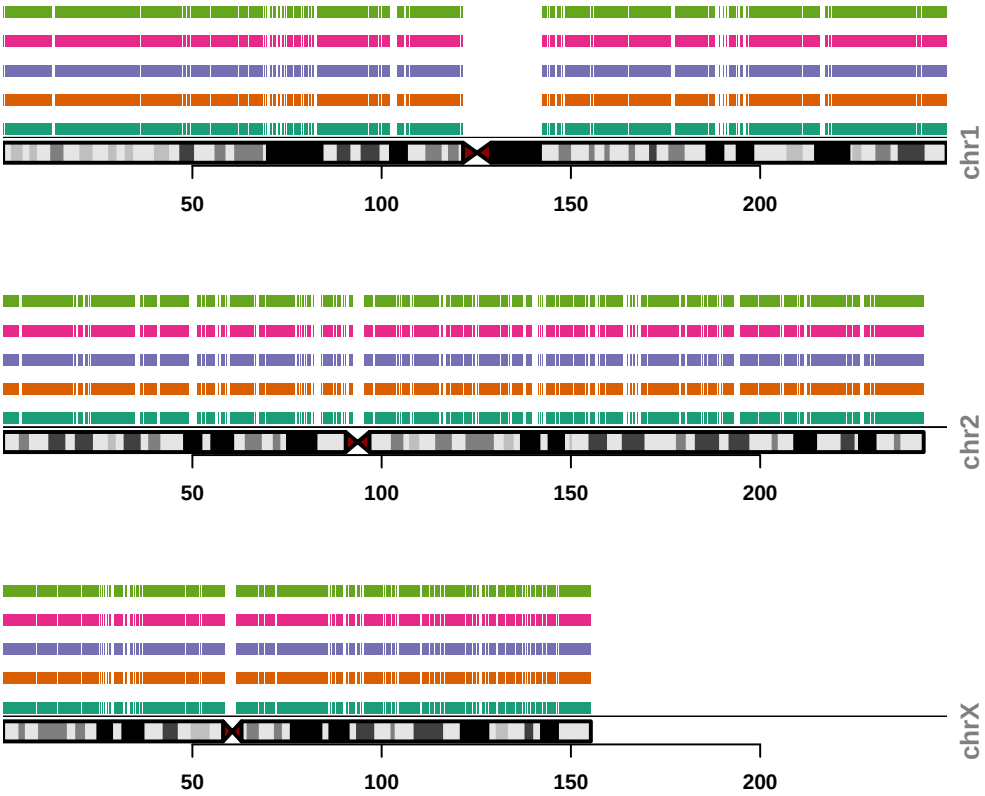
```
Plot chromosome done
Plot chromosome done
Plot chromosome done
```



4 Example 5: Plotting a set of coordinates as tracks

Here we plot multiple GRanges(), each as its own track.

```
> ideo_hg19 <- getIdeo("hg19")
> x <- GSM733664_broadPeaks
> gr <- GRanges(x[,1],IRanges(x[,2],x[,3]))
> pal <- brewer.pal(n=5,name="Dark2")
> chroms <- c("chr1","chr2","chrX")
> gr <- gr[which(seqnames(gr)%in% chroms)]
> chrom_bins <- getBins(chroms, ideo_hg19,
+                                          stepSize=5*100*1000)
> grList <- list(gr,gr,gr,gr,gr)
> plotOnIdeo(chrom=seqlevels(chrom_bins),
+           ideoTable=ideo_hg19,
+           values_GR=grList, value_cols="value",
+           plotType="seg_tracks",
+           col=pal, vertical=F)
```

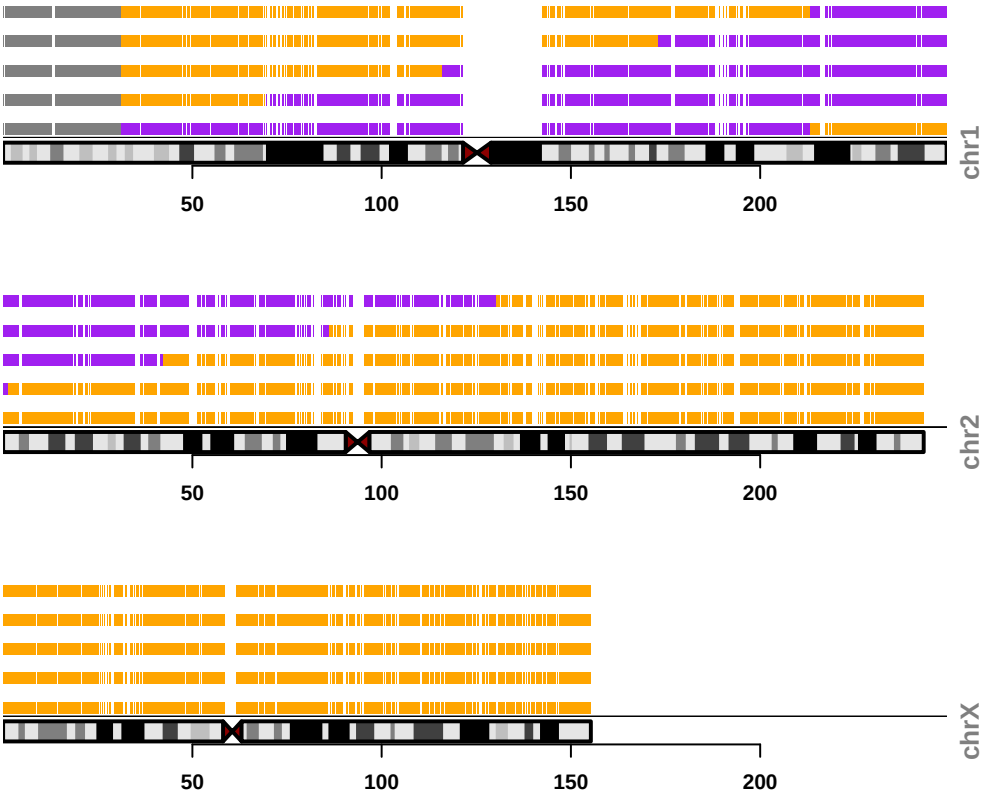


Segments can also be colour-coded by group type. For this the GRanges object needs to have a metadata column named "group", which has the pre-defined categories

```
> # assign group categories
> for (k in 1:5) {
+     gp <- rep("type1",length(grList[[k]]));
+     gp[(k*1000):((k*1000)+4000)] <- "type2"
+     gp[1:1000] <- "type3"
+     grList[[k]]$group <- gp
+     print(table(grList[[k]]$group))
+ }
```

```
type1 type2 type3
 9825  4000  1000

type1 type2 type3
 9824  4001  1000

type1 type2 type3
 9824  4001  1000

type1 type2 type3
 9824  4001  1000

type1 type2 type3
 9824  4001  1000
```

```
> # notice we don't name type3 - this is to show behaviour if a name is not specified
> namedCols <- c("orange","purple"); names(namedCols) <- c("type1","type2")
> plotOnIdeo(chrom=seqlevels(chrom_bins), ideoTable=ideo_hg19,values=grList,
+            plotType="seg_tracks",col=namedCols,vertical=F)
```



Session info

```
> sessionInfo()

R version 4.1.0 RC (2021-05-10 r80283)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Mojave 10.14.6

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
[1] IdeoViz_1.28.0       rtracklayer_1.52.0   RColorBrewer_1.1-2
[4] GenomicRanges_1.44.0 GenomeInfoDb_1.28.0  IRanges_2.26.0
[7] S4Vectors_0.30.0     Biobase_2.52.0       BiocGenerics_0.38.0

loaded via a namespace (and not attached):
 [1] rstudioapi_0.13             XVector_0.32.0
 [3] zlibbioc_1.38.0             GenomicAlignments_1.28.0
 [5] BiocParallel_1.26.0         lattice_0.20-44
```

```
 [7] rjson_0.2.20               tools_4.1.0
 [9] grid_4.1.0                 SummarizedExperiment_1.22.0
[11] matrixStats_0.58.0         yaml_2.2.1
[13] crayon_1.4.1               BiocIO_1.2.0
[15] Matrix_1.3-3               GenomeInfoDbData_1.2.6
[17] restfulr_0.0.13            bitops_1.0-7
[19] RCurl_1.98-1.3             DelayedArray_0.18.0
[21] compiler_4.1.0             MatrixGenerics_1.4.0
[23] Biostrings_2.60.0          Rsamtools_2.8.0
[25] XML_3.99-0.6
```