Introduction to the plotAlongChrom function

zhenyu

October 24, 2023

1 Introduction to the example data

The purpose of this vignette is to demonstrate some of the functionalities of the plotAlongChrom in the tilingArray package. We use a small subset data from an expression profiling paper [1]; The data only include the region from 35000bp to 50000bp in yeast chromosome one. Expression profiling is done in YPE and YPD conditions, 3 replicates each. Further information about the experimental design can be found at the paper website http://steinmetzlab.embl.de/NFRsharing/.

```r
> library("grid")
> library("RColorBrewer")
> library("tilingArray")

> data("segnf")
> class(segnf)

[1] "environment"

> ls(segnf)

[1] "1.+" "1.-"

> segnf$"1.+"

Object of class 'segmentation':
Data matrix: 1775 x 6
Change point estimates for number of segments S = 1:17
Confidence intervals for 1 fits from S = 17 to 17
Selected S = 17

> head(segnf$"1.+"@y)

                 YPE1  YPE2  YPE3  YPD1  YPD2  YPD3
[1,] -1.55  -2.70  -2.52  -4.78  -6.66  -4.62
[2,] -1.95  -4.20  -3.55  -3.93  -4.75  -4.74
[3,] -2.24  -2.24  -1.86  -6.15  -4.74  -3.82
[4,] -2.59  -2.52  -2.39  -4.31  -4.44  -4.40
[5,] -2.62  -3.91  -4.41  -4.94  -5.19  -4.65
[6,] -4.30  -4.82  -4.61  -5.62  -5.14  -5.60

> dim(segnf$"1.+"@y)

[1] 1775     6
```
The `segnf` object is an environment which contains two objects of class `segmentation`. `segnf` is the output of the `segChrom` in the `tilingArray` package. The `segmentation` object in `segnf` stores the probe expression information in the slot `y`. As can be seen, it contains 1775 probes and 6 array hybridizations in two conditions. The genomic coordinates where the probes aligned to is stored in the slot `x`. The order of the slot `x` is the same as the probe row order in slot `y`. The segment boundary information is stored in the slot `breakpoints` which is a list that contains all the optimal placement of 1 segment to the designated number (here in this data set is 17) of segments for this data. A log likelihood score for each placement is stored in slot `logLik` from which the best one is chosen and stored in the slot `nrSegments`. Further information about how the segmentation algorithm works, please read the vignette segmentation demo.

> data(gffSub)
> head(gffSub)

<table>
<thead>
<tr>
<th>id</th>
<th>chr</th>
<th>start</th>
<th>end</th>
<th>strand</th>
<th>source</th>
<th>feature</th>
<th>Name</th>
<th>orf_classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>41</td>
<td>35156</td>
<td>36304</td>
<td>+</td>
<td>SGD</td>
<td>gene</td>
<td>YAL060W</td>
<td>Verified</td>
</tr>
<tr>
<td>42</td>
<td>42</td>
<td>35156</td>
<td>36304</td>
<td>+</td>
<td>SGD</td>
<td>CDS</td>
<td>YAL060W</td>
<td>Verified</td>
</tr>
<tr>
<td>43</td>
<td>43</td>
<td>36497</td>
<td>36919</td>
<td>-</td>
<td>SGD</td>
<td>gene</td>
<td>YAL059C-A</td>
<td>Dubious</td>
</tr>
<tr>
<td>44</td>
<td>44</td>
<td>36497</td>
<td>36919</td>
<td>-</td>
<td>SGD</td>
<td>CDS_dubious</td>
<td>YAL059C-A</td>
<td>Dubious</td>
</tr>
<tr>
<td>45</td>
<td>45</td>
<td>36510</td>
<td>37148</td>
<td>+</td>
<td>SGD</td>
<td>gene</td>
<td>YAL059W</td>
<td>Verified</td>
</tr>
<tr>
<td>46</td>
<td>46</td>
<td>36510</td>
<td>37148</td>
<td>+</td>
<td>SGD</td>
<td>CDS</td>
<td>YAL059W</td>
<td>Verified</td>
</tr>
</tbody>
</table>

gene

41 BDH1
42 BDH1
43 <NA>
44 <NA>
45 ECM1
46 ECM1

The `gffSub` object is a data frame that contains the SGD annotated features of the region 35000bp-50000bp for yeast chromosome one.
2 Visualizing the expression profiling with the `plotAlongChrom` function

The function `plotAlongChrom` accepts an environment as its first argument, which is expected to contain objects of class `segmentation` with names given by `paste(chr, c("+", "-"), sep= ".")`, where `chr` is the chromosome identifier.

The following code generates Figure 1, a dot plot that averaged across all hybes.

```r
> grid.newpage()
> plotAlongChrom(segnf, chr=1, coord=c(35000, 50000), what="dots", gff=gffSub)
```

![Figure 1: Along-chromosome dot plot of the averaged value across all hybes.](image)

We could also make separate dot plot for different hybes by setting the parameter `sepPlot` as TRUE. The following code generates Figure 2 that plots the expression separately for the two conditions.

```r
> segObj = new.env(parent = baseenv())
> nmLabel = colnames(segnf$"1.+"@y)
> lab = gsub("\d", "", nmLabel)
> for(nm in paste(1, c("+", "-"), sep=".")){
+   s = get(nm, env = segnf)
+   rpY = tapply(1:length(lab), lab, function(i) rowMeans(s@y[,i]))
+   s@y = do.call(cbind, rpY)
+   assign(nm, s, segObj)
+ }
> grid.newpage()
> plotAlongChrom(segObj, chr=1, coord=c(35000, 50000), what="dots", gff=gffSub, sepPlot = T)
```

However, with the number of hybes increases, it is very hard to see the difference in dot plots in a normal screen. Thus, if the number of hybes is more than 4, the function will force to take the average. A better alternative of displaying multiple hybes is to use the heatmap. The following code generates Figure 3 that makes the heatmap plot.

```r
> grid.newpage()
> plotAlongChrom(segnf, chr=1, coord=c(35000, 50000), what="heatmap", gff=gffSub, sepPlot = TRUE, rowNamesHeatmap = nmLabel, makeRasterImage = FALSE)
```
Figure 2: Along-chromosome dot plot of the averaged value among different replicates for YPD and YPE condition.

Figure 3: Along-chromosome heatmap plot of all the replicates in YPD and YPE condition.
Start with R 2.11.0, the *grid* package introduced the raster array image function `grid.raster` which is a faster and efficient way of generating heatmap images. From R 2.11.0, The *tilingArray* package will use the `grid.raster` function as default to make heatmap images replacing the previous `grid.rect` function. The choice between the two drawing functions can be changed by the parameter `makeRasterImage`. The following code generates Figure 4 that makes the raster heatmap plot.

```r
> grid.newpage()
> plotAlongChrom(segnf, chr=1, coord=c(35000,50000), what="heatmap", gff=gffSub,
+     rowNamesHeatmap=nmLabel, makeRasterImage=TRUE)
```

![Figure 4: Along-chromosome raster heatmap plot of all the replicates in YPD and YPE condition.](image)

The color gradient of the heatmap could be changed by the parameter `colHeatmap`. The following code generates Figure 5 that makes the raster heatmap plot using a blue color gradient.

```r
> grid.newpage()
> plotAlongChrom(segnf, chr=1, coord=c(35000,50000), what="heatmap", gff=gffSub,
+     rowNamesHeatmap=nmLabel, makeRasterImage=TRUE,
+     colHeatmap = colorRamp(brewer.pal(9, "Blues")))
```

**References**

Figure 5: Along-chromosome raster heatmap plot of all the replicates in YPD and YPE condition with a blue color gradient.