ChromHeatMap

Tim F. Rayner

Cambridge Institute of Medical Research

April 30, 2024

1 Introduction

The **ChromHeatMap** package provides functions for visualising expression data in a genomic context, by generating heat map images in which data is plotted along a given chromosome for all the samples in a data matrix.

These functions rely on the existence of a suitable **AnnotationDbi** package which provides chromosome location information for the probe- or gene-level identifiers used in your data set. The data themselves must be in either an ExpressionSet, or a data matrix with row names corresponding to probe or gene identifiers and columns corresponding to samples. While the **ChromHeatMap** package was originally designed for use with microarray data, given an appropriate **AnnotationDbi** package it can also be used to visualise data from next-generation sequencing experiments.

The output heatmap can include sample clustering, and data can either be plotted for each strand separately, or both strands combined onto a single heat map. An idiogram showing the cytogenetic banding pattern of the chromosome will be plotted for supported organisms (at the time of writing: *Homo sapiens*, *Mus musculus* and *Rattus norvegicus*; please contact the maintainer to request additions).

Once a heat map has been plotted, probes or genes of interest can be identified interactively. These identifiers may then be mapped back to gene symbols and other annotation via the **AnnotationDbi** package.

2 Data preparation

Expression data in the form of a data matrix must initially be mapped onto its corresponding chromosome coordinates. This is done using the `makeChrStrandData`:

```
> library('ALL')
> data('ALL')
> selSamples <- ALL$mol.biol %in% c('ALL1/AF4', 'E2A/PBX1')
> ALLs <- ALL[, selSamples]
> library('ChromHeatMap')
> chrdata<-makeChrStrandData(exprs(ALLs), lib='hgu95av2')
```

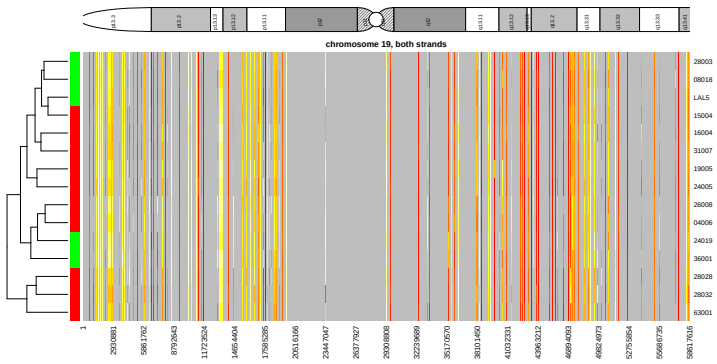
The output *chrdata* object here contains the expression data indexed by coordinate. Note that the `makeChrStrandData` function is based on the `Makesense` function in the **geneplotter** package, removing the internal call to lowess to avoid smoothing the data (which is undesirable in this case). The `makeChrStrandData` function is used specifically because it incorporates information on both the start and end chromosome coordinates for each locus. This allows the `plotChrMap` function to accurately represent target widths on the chromosome plot.

3 Plotting the heat map

Once the data has been prepared, a single call to `plotChrMap` will generate the chromosome heat map. There are many options available for this plot, and only a couple of them are illustrated here. Here we generate a whole-chromosome plot (chromosome 19), with both strands combined into a single heat map:

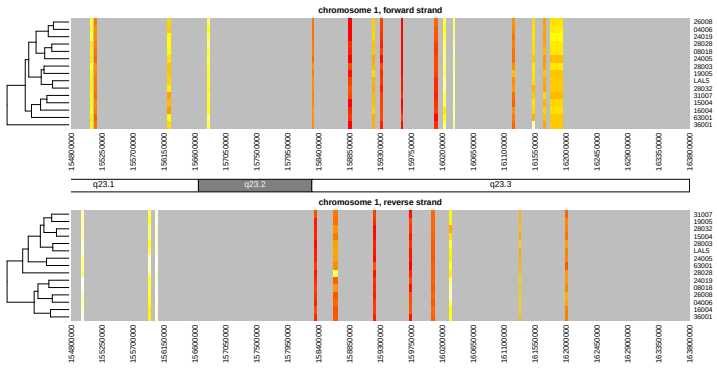
```
> groupcol <- ifelse( ALLs$mol.biol == 'ALL1/AF4', 'red', 'green' )
> plotChrMap(chrdata, 19, strands='both', RowSideColors=groupcol)
```

```
ChrMapPlot
Number of features plotted: 157
```



Chromosomes can be subsetted by cytoband or start/end coordinates along the chromosome. The following illustrates how one might plot the strands separately (this is the default behavior):

```
> plotmap<-plotChrMap(chrdata, 1, cytoband='q23', interval=50000, srtCyto=0, cexCyto=1.2)
```



Other options include subsetting of samples, adding a color key to indicate sample subsets, deactivating the sample-based clustering and so on. See the help pages for `plotChrMap` and `drawMapDendro` for details.

Note that the default colors provided by the `heat.colors` function are not especially attractive or informative; consider using custom-defined colors, for example by using the **RColorBrewer** package.

The output of the `plotChrMap` function can be subsequently used with the `grabChrMapProbes` function which enables the user to identify the probes or genes responsible for heatmap bands of interest.

Note that the `layout` and `par` options for the current graphics device are *not* reset following generation of the image. This is so that the `grabChrMapProbes` function can accurately identify the region of interest when the user interactively clicks on the diagram.

4 Interactive probe/gene identification

Often it will be of interest to determine exactly which probes or genes are shown to be up- or down-regulated by the `plotChrMap` heat map. This can be done using the `grabChrMapProbes` function. This takes the output of the `plotChrMap` function, asks the user to mouse-click the heatmap on either side of the bands of interest and returns a character vector of the locus identifiers in that region. These can then be passed to the **AnnotationDbi** function `mget` to identify which genes are being differentially expressed.

```
> probes <- grabChrMapProbes( plotmap )
> genes <- unlist(mget(probes, envir=hgu95av2SYMBOL, ifnotfound=NA))
```

Note that due to the way the expression values are plotted, genes which lie very close to each other on the chromosome may have been averaged to give a signal that could be usefully plotted at screen resolution. In such cases the locus identifiers will be returned concatenated, separated by semicolons (e.g. "`37687_i_at;37688_f_at;37689_s_at`"). Typically this is easily solved by zooming in on a region of interest, using either the "cytoband" or "start" and "end" options to `plotChrMap`. See also the "interval" option for another approach to this problem.

5 Session information

The version number of R and packages loaded for generating the vignette were:

```
R version 4.4.0 beta (2024-04-15 r86425)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 22.04.4 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.19-bioc/R/lib/libRblas.so
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

locale:
```

```
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: America/New_York
tzcode source: system (glibc)

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

other attached packages:
 [1] hgu95av2.db_3.13.0   org.Hs.eg.db_3.19.1  ChromHeatMap_1.58.0
 [4] annotate_1.82.0      XML_3.99-0.16.1      AnnotationDbi_1.66.0
 [7] IRanges_2.38.0       S4Vectors_0.42.0     ALL_1.45.0
[10] Biobase_2.64.0       BiocGenerics_0.50.0

loaded via a namespace (and not attached):
 [1] SparseArray_1.4.0           bitops_1.0-7
 [3] RSQLite_2.3.6               lattice_0.22-6
 [5] grid_4.4.0                  fastmap_1.1.1
 [7] blob_1.2.4                  jsonlite_1.8.8
 [9] Matrix_1.7-0                restfulr_0.0.15
[11] GenomeInfoDb_1.40.0         DBI_1.2.2
[13] httr_1.4.7                  UCSC.utils_1.0.0
[15] Biostrings_2.72.0           codetools_0.2-20
[17] abind_1.4-5                 cli_3.6.2
[19] rlang_1.1.3                 crayon_1.5.2
[21] XVector_0.44.0              bit64_4.0.5
[23] DelayedArray_0.30.0         cachem_1.0.8
[25] yaml_2.3.8                  S4Arrays_1.4.0
[27] tools_4.4.0                 parallel_4.4.0
[29] BiocParallel_1.38.0         memoise_2.0.1
[31] GenomeInfoDbData_1.2.12     Rsamtools_2.20.0
[33] SummarizedExperiment_1.34.0 curl_5.2.1
[35] vctrs_0.6.5                 R6_2.5.1
[37] png_0.1-8                   BiocIO_1.14.0
[39] matrixStats_1.3.0           rtracklayer_1.64.0
[41] zlibbioc_1.50.0             KEGGREST_1.44.0
[43] bit_4.0.5                   pkgconfig_2.0.3
[45] GenomicRanges_1.56.0        GenomicAlignments_1.40.0
[47] MatrixGenerics_1.16.0       xtable_1.8-4
[49] rjson_0.2.21                compiler_4.4.0
[51] RCurl_1.98-1.14
```