

ChromHeatMap

Tim F. Rayner

October 29, 2019

Cambridge Institute of Medical Research

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.

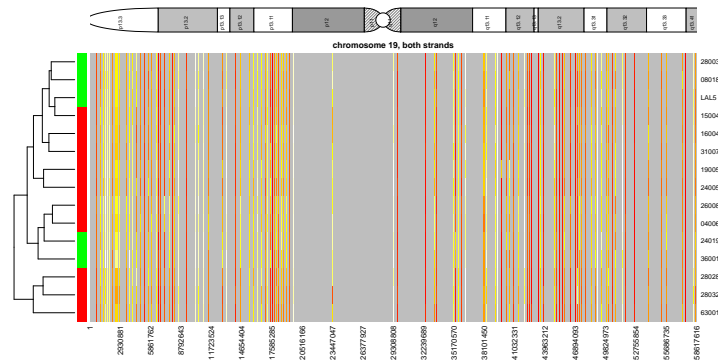
3 Plotting the heat map

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)
```

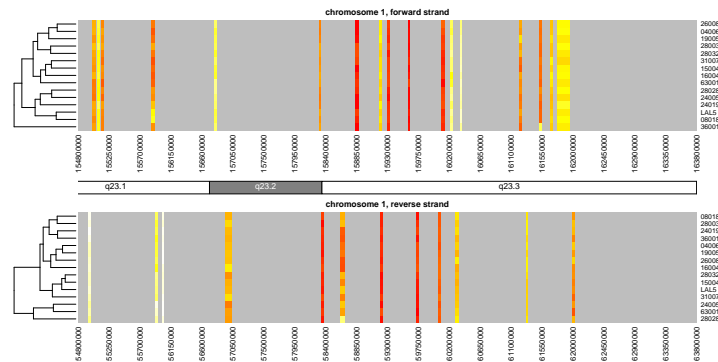
ChrMapPlot

Number of features plotted: 152



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 3.6.1 (2019-07-05)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows Server 2012 R2 x64 (build 9600)
```

```
Matrix products: default
```

```
locale:
[1] LC_COLLATE=C
[2] LC_CTYPE=English_United States.1252
```

```
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252
```

attached base packages:

```
[1] stats4      parallel  stats      graphics  grDevices  utils      datasets
[8] methods     base
```

other attached packages:

```
[1] hgu95av2.db_3.2.3      org.Hs.eg.db_3.10.0  ChromHeatMap_1.40.0
[4] annotate_1.64.0        XML_3.98-1.20        AnnotationDbi_1.48.0
[7] IRanges_2.20.0        S4Vectors_0.24.0    ALL_1.27.0
[10] Biobase_2.46.0        BiocGenerics_0.32.0
```

loaded via a namespace (and not attached):

```
[1] Rcpp_1.0.2              pillar_1.4.2
[3] compiler_3.6.1          GenomeInfoDb_1.22.0
[5] XVector_0.26.0          bitops_1.0-6
[7] tools_3.6.1             zlibbioc_1.32.0
[9] zeallot_0.1.0           digest_0.6.22
[11] bit_1.1-14              lattice_0.20-38
[13] RSQLite_2.1.2           memoise_1.1.0
[15] tibble_2.1.3            pkgconfig_2.0.3
[17] rlang_0.4.1             Matrix_1.2-17
[19] DelayedArray_0.12.0     DBI_1.0.0
[21] GenomeInfoDbData_1.2.2  rtracklayer_1.46.0
[23] Biostrings_2.54.0       vctrs_0.2.0
[25] grid_3.6.1             bit64_0.9-7
[27] BiocParallel_1.20.0     blob_1.2.0
[29] matrixStats_0.55.0      backports_1.1.5
[31] Rsamtools_2.2.0         GenomicRanges_1.38.0
[33] GenomicAlignments_1.22.0 SummarizedExperiment_1.16.0
[35] xtable_1.8-4            RCurl_1.95-4.12
[37] crayon_1.3.4
```