

ChromHeatMap

Tim F. Rayner

October 30, 2017

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.

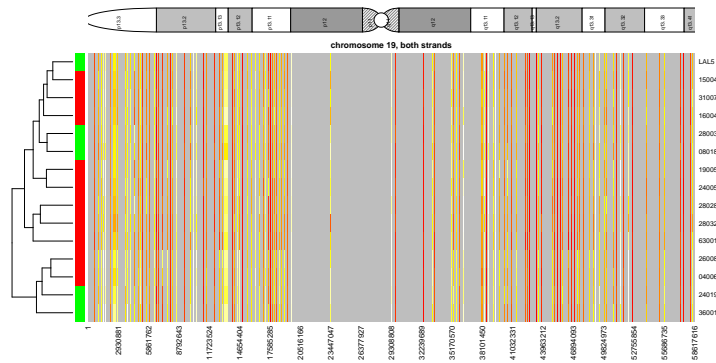
3 Plotting the heat map

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)
```

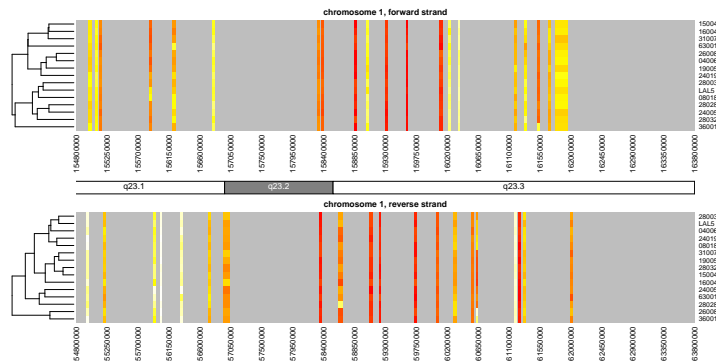
ChrMapPlot

Number of features plotted: 167



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 3.4.2 (2017-09-28)
Platform: x86_64-apple-darwin15.6.0 (64-bit)
Running under: OS X El Capitan 10.11.6

Matrix products: default
BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/3.4/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] hgu95av2.db_3.2.3      org.Hs.eg.db_3.4.2      ChromHeatMap_1.32.0
[4] annotate_1.56.0         XML_3.98-1.9            AnnotationDbi_1.40.0
[7] IRanges_2.12.0         S4Vectors_0.16.0       ALL_1.19.0
[10] Biobase_2.38.0         BiocGenerics_0.24.0
```

loaded via a namespace (and not attached):

```
[1] Rcpp_0.12.13           compiler_3.4.2
[3] GenomeInfoDb_1.14.0    XVector_0.18.0
[5] bitops_1.0-6           tools_3.4.2
[7] zlibbioc_1.24.0        digest_0.6.12
[9] bit_1.1-12             RSQLite_2.0
[11] memoise_1.1.0          tibble_1.3.4
[13] lattice_0.20-35        pkgconfig_2.0.1
[15] rlang_0.1.2            Matrix_1.2-11
[17] DelayedArray_0.4.0     DBI_0.7
[19] GenomeInfoDbData_0.99.1 rtracklayer_1.38.0
[21] Biostrings_2.46.0      bit64_0.9-7
[23] grid_3.4.2             BiocParallel_1.12.0
[25] blob_1.1.0             Rsamtools_1.30.0
[27] matrixStats_0.52.2     GenomicRanges_1.30.0
[29] GenomicAlignments_1.14.0 SummarizedExperiment_1.8.0
[31] xtable_1.8-2           RCurl_1.95-4.8
```