compEpiTools

Mattia Pelizzola, Kamal Kishore

October 24, 2023

Contents

1 Overview

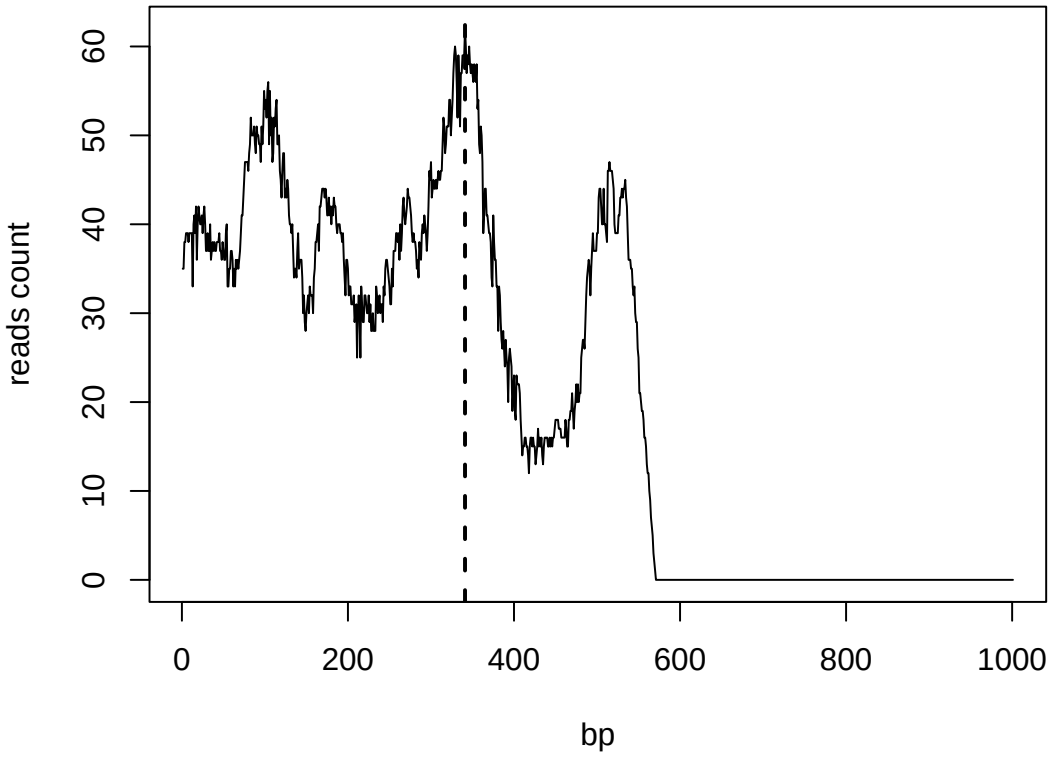
In this document you can find a brief tutorial on the *compEpiTools* package, a toolkit for the integrative analysis of epigenomics data, which can be complemented with the *methylPipe* package to include support on DNA methylation data. *compEpiTools* offers multiple functionalities covering five main areas: Counting reads in genomic regions, Annotation of genomic regions, Functional annotation, Visualization, and Other (see ?compEpiTools at the R prompt for a brief overview). Many of these methods and functions concern topics of interest in epigenomics, including: identification of enhancer and matching with putative target regions, indentification of long non coding RNAs based on chromatin features, computation of PolII stalling index, determination of promoter CpG content etc. Finally, functionalities are avaliable to integrate and display heterogenous data-types across multiple genomic regions.

2 Counting reads in GRanges

These are mostly *GRanges* methods facilitating several common operations concerning **counting** in genomic regions aligned reads stored in BAM files. In this example a small BAM file is used to compute the base-level coverage using `GRbaseCoverage`. Next the count of reads is determined for a whole genomic region (`GRcoverage`) and dividing it in 5 equally sized bins (`GRcoverageInbins`). Finally the summit, i.e. the position of maximum coverage is identified and highlighted in the plot of base-level coverage for one of these regions.

```
library(compEpiTools)
require(TxDb.Mmusculus.UCSC.mm9.knownGene)
require(org.Mm.eg.db)
txdb <- TxDb.Mmusculus.UCSC.mm9.knownGene
```

```
bampath <- system.file("extdata", "ex1.bam", package="Rsamtools")
ir <- IRanges(start=c(1000, 100), end=c(2000, 1000))
gr <- GRanges(seqnames=Rle(c('seq1','seq2')), ranges=ir)
res <- GRbaseCoverage(Object=gr, bam=bampath)
                        ## Warning:  'applyPileups' is deprecated.
                                 ## Use 'pileup' instead.
                                ## See help("Deprecated")

GRcoverage(Object=gr, bam=bampath, Nnorm=FALSE, Snorm=FALSE)[1]

                        ## Warning:  'applyPileups' is deprecated.
                                 ## Use 'pileup' instead.
                                ## See help("Deprecated")

## [1] 20359

GRcoverageInbins(Object=gr, bam=bampath, Nnorm=FALSE, Snorm=FALSE, Nbins=5)[1,]

                        ## Warning:  'applyPileups' is deprecated.
                                 ## Use 'pileup' instead.
                                ## See help("Deprecated")

## [1] 8148 7969 4242    0    0

summit <- GRcoverageSummit(Object=gr, bam=bampath)

                        ## Warning:  'applyPileups' is deprecated.
                                 ## Use 'pileup' instead.
                                ## See help("Deprecated")

plot(res[[1]], type='l', xlab='bp', ylab='reads count')
abline(v=start(summit[1])-start(gr[1])+1, lty=2, lwd=2)
```



Apart from these general functionalities, two specific applications of interest in epigenetic are the determination of ChIP-seq enrichment and of the PolII stalling index.

In (epi)genomics ChIP-seq is an experimental method to identify the genomic regions which are bound by a DNA binding protein (if any), such as transcription factor or an histone mark. Such experiments are typically conducted comparing a sample in which an antibody specific for the target protein was used to enrich for the bound genomic DNA (ChIP), which is then compared to a control sample where the antibody was not used (input). Comparing the signal in the ChIP and input samples allows identifying regions of significant enrichment, the so called ChIP-seq peaks. The peaks can also be quantitatively scored to give a measure to the binding intensity of the factor of interest. The `GRenrichment` method can be used to determine this **enrichment**. First the number of reads in the ChIP sample within the peak is determined and normalized by the total number of reads aligned in the BAM file (ChIPw), then the same is done for the input sample (inputw). Finally the log2(ChIPw - inputw) is determined which was shown to be approximatively linearly correalated to the log of the qPCR enrichment, typically used for validation.

The **PolII stalling index** is a measure that is commonly determined to estimate the degree of stalling of the Polymeras II in the region around the Transcription Start Sites (TSS), which is somehow related to the elongation rate, the speed at which the PolII is actively elongating and transcribing the open reading frame. The stalling index is determined as the ratio of the PolII ChIP-seq signal in the TSS region compared to the gene body, and can be determined using the `stallingIndex` function and is typically visualized with cumulative plot generated by the `plotStallingIndex` function.

3 Annotation of genomic regions

These methods can be used to assign to genomic ranges annotations concerning gene annotation and user-defined databases. Specifically, the `TSS` and `distanceFromTSS` methods can be used to determine the Transcription Start Sites positions for all transcript in a *TranscriptDb* database and to compute the distance between a set of genomic regions and the most proximal TSS. The `GRangesInPromoters` is a convenience wrapper to subset a *GRanges* object keeping only ranges overlapping (or not) with gene promoter regions. All these methods plus `GRmidpoint` are used by `GRannotateSimple` to partition a set of genomic ranges into those overlapping with promoters, intragenic and intergenic regions.

Apart from genes and promoters, numerous other genomic annotation resources are available for example in the UCSC table browser. These or other user-defined regions of interest, together with genes and promoter annotations can be used with the `GRannotate` method to obtain the complete annotation of a set of genomic ranges. This is illustrated in the following example, where CpG Islands (CGIs) are considered in addition to genes and promoters. In the output each genomic range is put in the context of the nearest TSS ('nearest' columns), the specific location in which it falls ('location' columns) and the overlap with CGIs:

```
TSSpos <- TSS(txdb)
gr <- TSSpos[1:5]
start(gr) <- start(gr) - 1000
end(gr) <- end(gr) - 600
mcols(gr) <- NULL
# retrieving CGI mm9 islands from UCSC annotation tables
cgipath <- system.file("extdata", "CGIgr_mm9.Rdata", package="compEpiTools")
load(cgipath)
res <- GRannotate(Object=GRmidpoint(gr), txdb=txdb, EG2GS=org.Mm.eg.db,
        upstream=2000, downstream=1000, userAnn=GRangesList(CGI=CGIgr_mm9))
show(res)

## GRanges object with 5 ranges and 9 metadata columns:
##         seqnames    ranges strand | nearest_tx_name distance_fromTSS
##            <Rle> <IRanges>  <Rle> |     <character>        <integer>
##   18777     chr1   4797174      + |      uc007afg.1              799
##   18777     chr1   4797174      + |      uc007afg.1              799
##   21399     chr1   4846975      + |      uc007afi.2              799
##   21399     chr1   4846975      + |      uc007afi.2              799
##   21399     chr1   4847609      + |      uc007afi.2              165
##         nearest_gene_id nearest_gene_symbol              location
##             <character>         <character>            <character>
##   18777           18777              Lypla1      promoter;promoter
##   18777           18777              Lypla1      promoter;promoter
##   21399           21399               Tcea1 promoter;promoter;pr..
##   21399           21399               Tcea1 promoter;promoter;pr..
##   21399           21399               Tcea1 promoter;promoter;pr..
##                location_tx_id  location_gene_id location_gene_symbol
##                   <character>       <character>          <character>
##   18777  uc007afg.1;uc007afh.1       18777;18777        Lypla1;Lypla1
##   18777  uc007afg.1;uc007afh.1       18777;18777        Lypla1;Lypla1
##   21399 uc007afi.2;uc011wht... 21399;21399;21399    Tcea1;Tcea1;Tcea1
##   21399 uc007afi.2;uc011wht... 21399;21399;21399    Tcea1;Tcea1;Tcea1
```

```
##   21399 uc007afi.2;uc011wht... 21399;21399;21399    Tcea1;Tcea1;Tcea1
##                CGI
##          <numeric>
##   18777         0
##   18777         0
##   21399         0
##   21399         0
##   21399         1
##   -------
##   seqinfo: 35 sequences (1 circular) from mm9 genome
```

The `makeGtfFromDb` method can be used to export a gene or transcript level GTF file to be used with standard aligners, such as TopHat or reads counting tools, such as the one available in HTSeq, in order to be consistent with analyses performed within R and Bioconductor.

4 Functional annotation

A number of functions and methods is available to define epigenetically relevant functional annotations. The `enhancers` allows pointing to putative enhancers based on H3K4me1 (thus pointing to enhancers which could be either active or poised) or H3K27ac (thus pointing to active enhancers) marks. To this purpose, distal peaks of these marks laying outside gene promoters are identified, and required to not overlap with CpG Islands to avoid peaks matching to potentially unannotated promoter regions.

Using the `matchEnhancers` it is possible to match enhancers with putative targets sites. Target sites could either be TSS or transcription factor binding sites localized at the level of promoters. Constraints are given based on a minimum or maximum distance between the enhancer and the target site. At the same time no additional TSS have to be present in between the enhancer and the putative target site (identification of 'direct' enhancers). This does not apply if those TSS belong to isoforms of the same gene. This method returns: (i) a set of reference regions without any interacting direct enhancers, (ii) a set of enhancers sites having putative taget regions, and (iii) those of putative target regions under control of enhancer sites. Lists (ii) and (iii) are ordered so that they can be immediately matched. Finally, if TF binding is provided, these two lists will be further divided considering only TF-bound TSS (target regions) which are bound by to enhancer bound from the same TF or not. This could set the foundation for further explorative analyses on nextworks of enhancers and target regions, possibly as a function of the binding of a TF.

`topGOres` and `simplifyGOterms` are convenience functions to deal to GeneOntology enrichment analyses. In particular the latter can be used to keep only the most informative GO terms. This is based on the fact that GeneOntology is composed of three different ontologies (Biological Processes, Molecular Functions and Cellular Components). Within each ontology, a set of GO terms describing those categories are available, together with the relationships linking them. Terms specifying more precisely a biological category are called children (e.g. induction of apopotosis) of more generic parent terms (e.g. apopotosis). Most informative GO terms to keep are defined here as those terms for which an enriched children term mapping to a very similar set of genes has not been also identified. If that happens, the children term is believed to contain most of the information, and typically better specifies the enrichmed GO category, comprared to the more general, less specific, parent term which are thus discarded.

`findLncRNA` is a function to point to putative intergenic long non-coding RNAs (lncRNAs). These are typically identifed thanks to their epigenetic signatures, characteristic of transcriptional units unrelated from gene-coding ones, associated to known genes. For simplicity, this function can only point to lncRNAs which are distal from gene transcriptional units, to avoid False Positives due to overlap with coding transcriptional units. While the detection is mostly based on H3K4me3 and H3K4me1 peaks,

additional marks and RNA-seq data can be used to have a more robust lncRNAs identification. See the documentation of the `findLncRNA` function for details.

`getPromoterClass` can be used to classify promoters according to their CpG content. In fact, it has been show that promoters with significantly different CpG content can be differently responsive to the presence/absence or even to the level of epigenetic marks such as DNA-methylation (Koga et al, Genome Research 2009).

5 Visualization

`heatmapData` and the associated `heatmapPlot` are very flexible functions that can be used with a range of data types to associate counts for multiple data or annotation tracks to genomic regions of interst (ROIs), and suqsequently visualizing similarites in those patterns. Supported data types range from BAM files, *GRanges* objects, GRanges metadata, putative methylation sites and their associated absolute and relative methylation level. All these data types are highly relevant for epigenomics integrative analyses and can include but they are not limited to: base-resolution or low-resolution DNA methylation data, histone marks, transcription factor binding, RNA-seq expression, which can be freely combined (data tracks). Importantly, predefined or user-defined genomic annotation(s) (annotation tracks) could be overlayed to further stratify the patterns emerging from the data tracks.

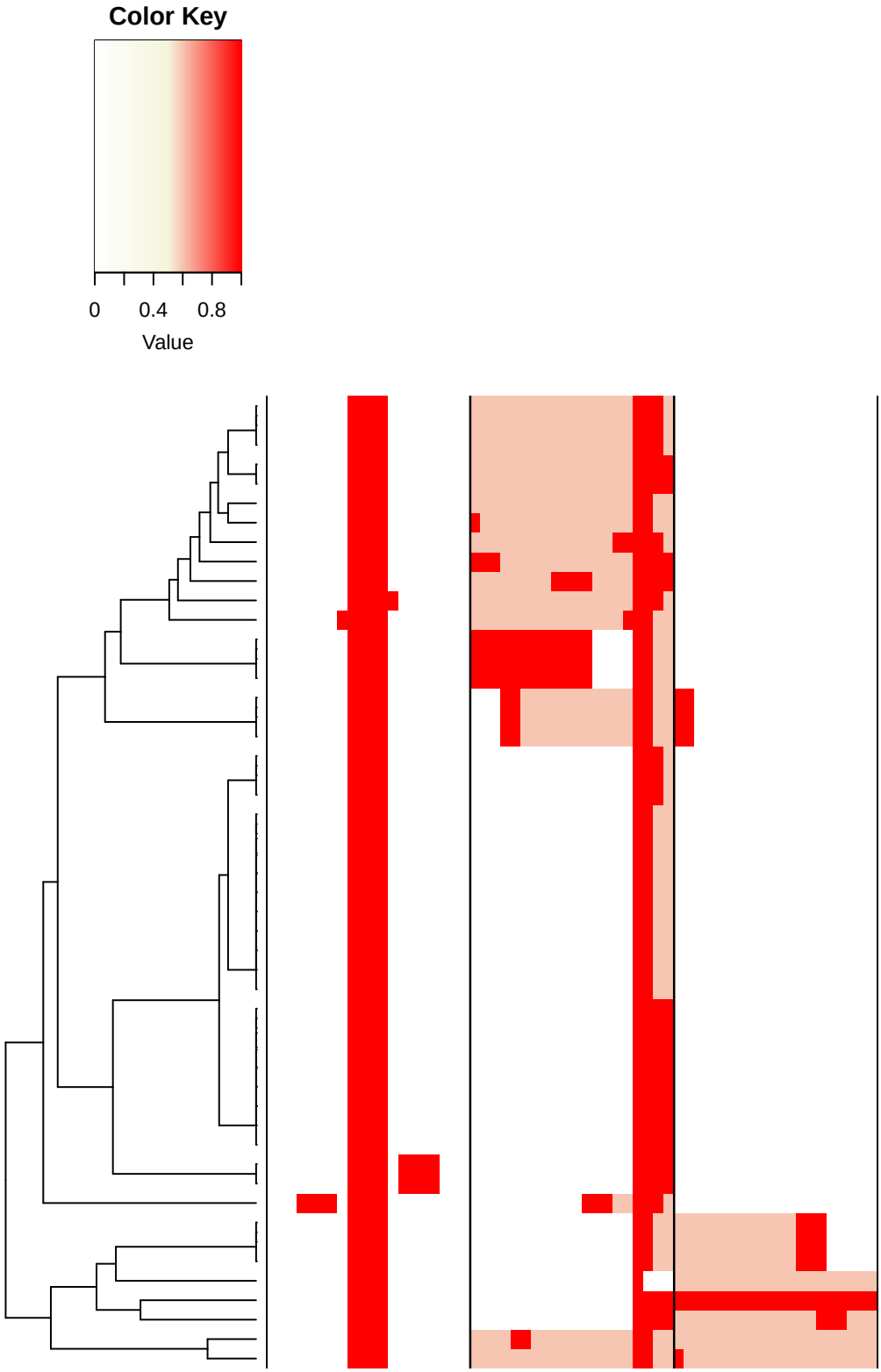
The counts associated to each given data track can be visualized using heatmaps, clustering regions with similar patterns over all (or a subset) of the provided data and annotation tracks (easily dozens of them, covering hundreads or thousands of genomic regions), together with the underlying gene annotation on the forward and reverse strand. ROIs can be potentially divided in a number of bins to provide higher resolution and information about patterns of the data within those regions.

BAM files can be provided to count reads in ROIs; these files could countain any kind of high-throughput sequencing data (including histone marks or transcription factor binding sites) in the case one wants to focus to the density of the reads on the genome, independently from the identification of significantly enriched/scoring regions. On the other hand *GRanges* can be provided to pass for example ChIP-seq peaks, or MeDIP-seq methylated regions, focusing on presence/absence of a given mark in the ROIs. The `mcols` of *GRanges* having th esame length of the ROIs could be also pre-populated by the user and used directly to fill in the relavant information of a given data track. DNA methylation relevant data could be provided using *GElist* or *GEcollection* object of the *methylPipe* package. All these data types can be freely combined in the input list of the `heatmapPlot` function.

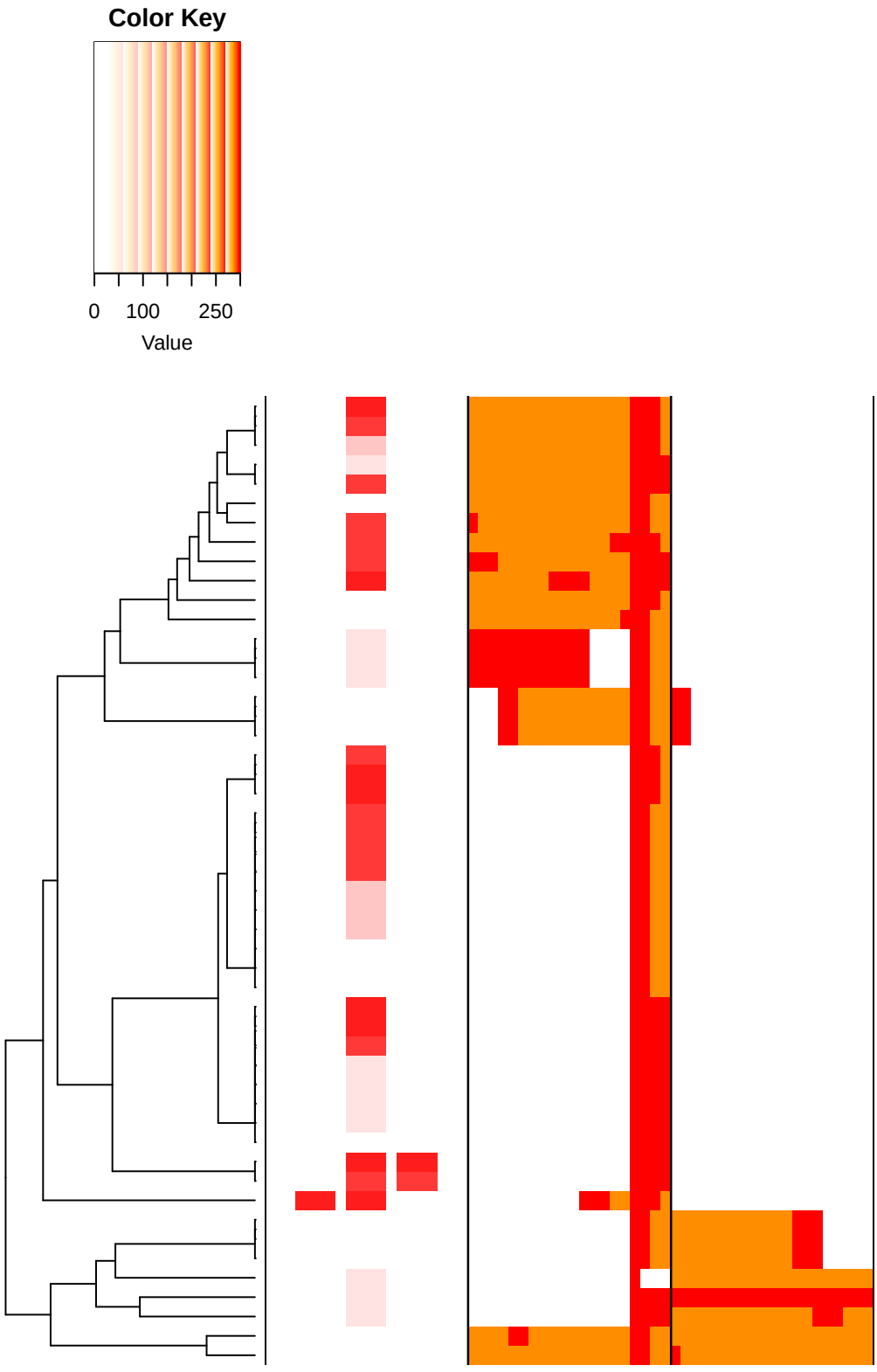
Importantly, the colorscale in the heatmap can otionally be set to display the significane of the data associated with any track in each genomic region. In the following example two very simple heatmaps are drawn, reporting putative transcription factor binding sites (actually TSS proximal regions) in the context of gene annotation (introns and exons are reported with dark and light red in the forward and reverse strand, respectively). In the following heatmap the same is displayed according to the p-value for each peak, where regions associated to high p-values are dimmed to a less intense colour.

```
gr <- TSSpos[1:50]
start(gr) <- start(gr) - 1000
end(gr) <- end(gr) - 600
extgr <- GRanges(seqnames(gr), ranges=IRanges(start(gr) - 1000, end(gr) + 1000))
data <- heatmapData(grl=list(ChIPseq=gr), refgr=extgr, type='gr', nbins=20, txdb=txdb)

## [1] "ChIPseq"

pvalues <- c(runif(20,1e-20,1e-8), runif(15,1e-4,1e-2), runif(15,0.5,1))
pvalues <- cbind(pvalues, rep(0, 50), rep(0, 50))
```

```
rownames(data[[1]][[1]]) <- paste(1:50, signif(pvalues[,1],1), sep=' # ')
heatmapPlot(matList=data[[1]], clusterInds=1:3)
```



```
heatmapPlot(matList=data[[1]], sigMat=pvalues, clusterInds=1:3)
```



6 Session Information

```
sessionInfo()

## R version 4.3.1 (2023-06-16)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 22.04.3 LTS
##
## Matrix products: default
## BLAS:   /home/biocbuild/bbs-3.18-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_GB              LC_COLLATE=C
##  [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
## [7] methods   base
##
## other attached packages:
##  [1] org.Mm.eg.db_3.18.0
##  [2] TxDb.Mmusculus.UCSC.mm9.knownGene_3.2.2
##  [3] GenomicFeatures_1.54.0
##  [4] compEpiTools_1.36.0
##  [5] GenomicRanges_1.54.0
##  [6] GenomeInfoDb_1.38.0
##  [7] topGO_2.54.0
##  [8] SparseM_1.81
##  [9] GO.db_3.18.0
## [10] AnnotationDbi_1.64.0
## [11] IRanges_2.36.0
## [12] S4Vectors_0.40.0
## [13] Biobase_2.62.0
## [14] graph_1.80.0
## [15] BiocGenerics_0.48.0
##
## loaded via a namespace (and not attached):
##   [1] RColorBrewer_1.1-3          rstudioapi_0.15.0
##   [3] magrittr_2.0.3              rmarkdown_2.25
##   [5] BiocIO_1.12.0               zlibbioc_1.48.0
##   [7] vctrs_0.6.4                 memoise_2.0.1
```

```
##   [9] Rsamtools_2.18.0           RCurl_1.98-1.12
##  [11] base64enc_0.1-3            htmltools_0.5.6.1
##  [13] S4Arrays_1.2.0             progress_1.2.2
##  [15] curl_5.1.0                 SparseArray_1.2.0
##  [17] Formula_1.2-5              KernSmooth_2.23-22
##  [19] htmlwidgets_1.6.2          Gviz_1.46.0
##  [21] cachem_1.0.8               GenomicAlignments_1.38.0
##  [23] lifecycle_1.0.3            pkgconfig_2.0.3
##  [25] Matrix_1.6-1.1             R6_2.5.1
##  [27] fastmap_1.1.1              GenomeInfoDbData_1.2.11
##  [29] MatrixGenerics_1.14.0      digest_0.6.33
##  [31] colorspace_2.1-0           Hmisc_5.1-1
##  [33] RSQLite_2.3.1              filelock_1.0.2
##  [35] fansi_1.0.5                httr_1.4.7
##  [37] abind_1.4-5                compiler_4.3.1
##  [39] bit64_4.0.5                marray_1.80.0
##  [41] htmlTable_2.4.1            backports_1.4.1
##  [43] BiocParallel_1.36.0        DBI_1.1.3
##  [45] highr_0.10                 methylPipe_1.36.0
##  [47] gplots_3.1.3               biomaRt_2.58.0
##  [49] rappdirs_0.3.3             DelayedArray_0.28.0
##  [51] rjson_0.2.21               gtools_3.9.4
##  [53] caTools_1.18.2             tools_4.3.1
##  [55] foreign_0.8-85             nnet_7.3-19
##  [57] glue_1.6.2                 restfulr_0.0.15
##  [59] grid_4.3.1                 checkmate_2.2.0
##  [61] cluster_2.1.4              generics_0.1.3
##  [63] gtable_0.3.4               BSgenome_1.70.0
##  [65] ensembldb_2.26.0           data.table_1.14.8
##  [67] hms_1.1.3                  xml2_1.3.5
##  [69] utf8_1.2.4                 XVector_0.42.0
##  [71] pillar_1.9.0               stringr_1.5.0
##  [73] limma_3.58.0               dplyr_1.1.3
##  [75] BiocFileCache_2.10.0       lattice_0.22-5
##  [77] deldir_1.0-9               rtracklayer_1.62.0
##  [79] bit_4.0.5                  biovizBase_1.50.0
##  [81] tidyselect_1.2.0           Biostrings_2.70.0
##  [83] knitr_1.44                 gridExtra_2.3
##  [85] ProtGenerics_1.34.0        SummarizedExperiment_1.32.0
##  [87] xfun_0.40                  statmod_1.5.0
##  [89] matrixStats_1.0.0          stringi_1.7.12
##  [91] lazyeval_0.2.2             yaml_2.3.7
##  [93] evaluate_0.22              codetools_0.2-19
##  [95] interp_1.1-4               tibble_3.2.1
##  [97] cli_3.6.1                  rpart_4.1.21
##  [99] munsell_0.5.0              Rcpp_1.0.11
## [101] dichromat_2.0-0.1          dbplyr_2.3.4
## [103] png_0.1-8                  XML_3.99-0.14
## [105] parallel_4.3.1             ggplot2_3.4.4
```

```
## [107] blob_1.2.4                  prettyunits_1.2.0
## [109] jpeg_0.1-10                 latticeExtra_0.6-30
## [111] AnnotationFilter_1.26.0     bitops_1.0-7
## [113] VariantAnnotation_1.48.0    scales_1.2.1
## [115] crayon_1.5.2                rlang_1.1.1
## [117] KEGGREST_1.42.0
```