ChIP-seq Data Analysis

Install R packages

In order to perform ChIP-seq data analysis with R, we have to first make sure necessary R packages installed before any processing. Bioconductor http://www.bioconductor.org is a large community which documents thousands of R packages dedicated to biomedical data analysis. We will pick several of those packages from Bioconductor for ChIP-seq data analysis. Note that some widely-used tools for ChIP-seq data analysis were not developed under R environment. Thus, R packages chosen here are for demonstration purpose, and might not be well adopted by community users.

```
## install necessary Bioconductor packages
options(repos = c(CRAN = "https://cran.r-project.org"))

if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
BiocManager::install()
install.packages('rmarkdown')
BiocManager::install("knitr") ## dealing with rmarkdown
BiocManager::install("GenomicRanges") ## dealing with genomic intervals
BiocManager::install("normr") ## dealing with peak calling
BiocManager::install("ChIPseeker") ## dealing with peak annotation
BiocManager::install("TxDb.Hsapiens.UCSC.hg38.knownGene") ## human genome and transcriptome
BiocManager::install("GenomeInfoDb")
BiocManager::install("BSgenome.Hsapiens.UCSC.hg38")
BiocManager::install("org.Hs.eg.db") ## gene name ontology
BiocManager::install("annotate") ## genome annotation
BiocManager::install("Rsamtools") ## dealing with BAM files
BiocManager::install("GenomicAlignments") ## dealing with sequencing reads
BiocManager::install("rtracklayer") ## genomic sequencing tracks
BiocManager::install("RColorBrewer") ## color code
```

After all required packages installed, we need to load these packages.

```
library(Rsamtools)
library(GenomicAlignments)
library(GenomicRanges)
library(normr)
library(TxDb.Hsapiens.UCSC.hg38.knownGene)
library(BSgenome.Hsapiens.UCSC.hg38)
library(org.Hs.eg.db)
library(ChIPseeker)
library(annotate)
library(rtracklayer)
library(RColorBrewer)
```

Goal of the lecture

We will demostrate how to analyze ChIP-seq data under R environment, from peak detection to peak annotation. We will summarize the characteristics of the given ChIP-seq dataset and report/visualize ChIP-seq profile related to individual genes.

Prepare ChIP-seq data

We will start from the mapped sequencing reads instead of the raw for two reasons: (1) our main practice here is to demonstrate ChIP-seq peak detection and annotation. (2) tools for raw data processing, *e.g.* reads mapping, are generally not developed under R. Mapped reads are usually stored with SAM format https://samtools.github.io/hts-specs/SAMv1.pdf. To shrink the size of the SAM files, SAM format is usually further converted to its binary version, *i.e.* BAM format. Basically, each record of SAM/BAM files represents one sequencing read. A ChIP-seq sequencing sample usually involves dozens of million of reads.

For demostration, we have generated small BAM files only containing sequencing reads on human chromosome 21 for protein CTCF binding in cell line GM12878 https://github.com/chingyaousf/Regulatory-Genomic-Sequencing-ChIP-seq-data-analysis/blob/main/data/. The full genome-wide sequencing data can be downloaded from here: https://www.encodeproject.org/experiments/ENCSR000AKB/. As we discussed before, we also need INPUT sample to control experimental noises: https://www.encodeproject.org/experiments/ENCSR000AKJ/. We can download both files with the following code and put all files into a folder for downstream analysis. Note, we also download the index files for the BAM files here.

```
## download the ChIP-seq sequencing reads, and put the files into a folder
## depending on the operating system, folder name might need to be changed.
workdir = '~/' ##
download.file('https://github.com/chingyaousf/Regulatory-Genomic-Sequencing-ChIP-seq-data-analysis/raw/r
download.file('https://github.com/chingyaousf/Regulatory-Genomic-Sequencing-ChIP-seq-data-analysis/raw/r
download.file('https://github.com/chingyaousf/Regulatory-Genomic-Sequencing-ChIP-seq-data-analysis/raw/r
download.file('https://github.com/chingyaousf/Regulatory-Genomic-Sequencing-ChIP-seq-data-analysis/raw/r
download.file('https://github.com/chingyaousf/Regulatory-Genomic-Sequencing-ChIP-seq-data-analysis/raw/r
```

Exploratory data analysis

Exploratory data analysis helps us better understand our data using simple techniques/operation.

With the downloaded datasets, we can compare ChIP and INPUT to see if ChIP actually contains more "peaks" than INPUT. To do so, we will use functions from R package **GenomicAlignments**. We first specify ChIP and INPUT file, separately, then count the sequencing reads in these files that are overlapped with consecutive genomic windows of 1000bp width. Here, for exploratory purpose, we use large window size to mimic peak regions. Then, we can apply simple normalization procedure based on library size to make data "comparable" across samples. Following code shows the read counts of first several windows on chr21 in both ChIP and INPUT.

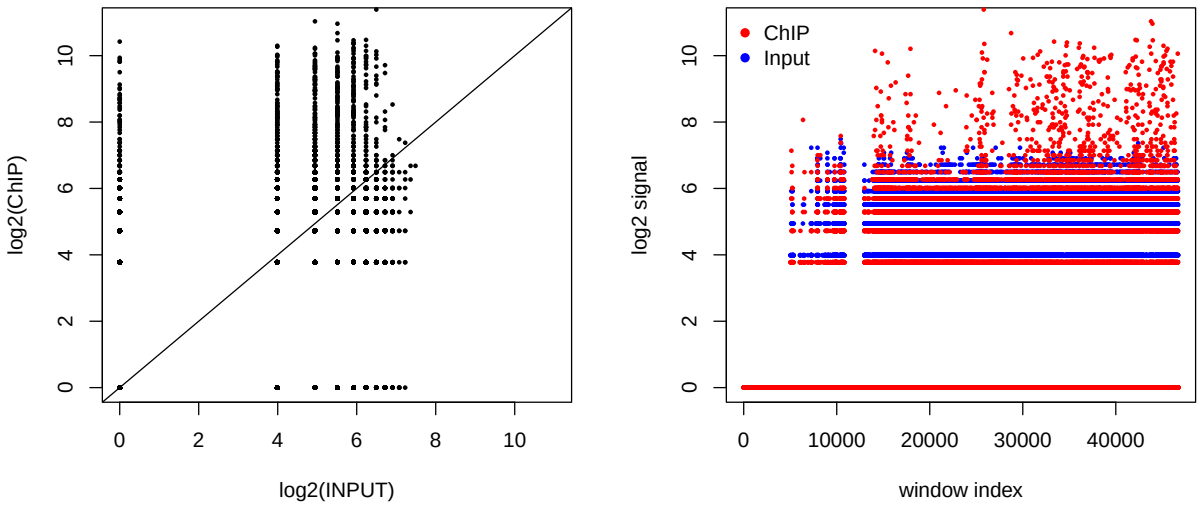
```
## sequencing files that only contain reads from chromosome 21
chip = file.path(workdir,'Gm12878Ctcf.bam')
input = file.path(workdir,'Gm12878Control.bam')
## chromosome length info from human genome build hg19
hg_chrs = getBSgenome("hg38")
seqlen_chr21 = seqlengths(hg_chrs)['chr21']
## create genomic 1000bp windows and store them using GenomicRanges
window_gr = unlist(tileGenome(seqlen_chr21, tilewidth=1000))
## count reads from both files for all windows
```

```
#rc = summarizeOverlaps(window_gr, c(chip, input))
#rc = assays(rc)[[1]]
load(paste0(workdir,'preloadforwindows.rda'))
## simple normalization based on library size
cpm_chr21 = t(t(rc)*(1000000/colSums(rc)))
head(cpm_chr21)
```

```
##      Gm12878Ctcf.bam Gm12878Control.bam
## [1,]               0                  0
## [2,]               0                  0
## [3,]               0                  0
## [4,]               0                  0
## [5,]               0                  0
## [6,]               0                  0
```

We will present the data intuitively with visualization: (1) which windows are different between ChIP and INPUT (2) where data is located on the genome. The first figure indicates the enrichment of reads in windows that are shown above the diagonal line; the second figure indicates physically where those enriched windows are located on the chromosome.

```
par(mfrow=c(1,2))
## read counts in all windows
plot(log2(cpm_chr21[,2]+1),log2(cpm_chr21[,1]+1),pch=16,cex=0.5,xlab='log2(INPUT)',ylab='log2(ChIP)',yli
abline(a=0,b=1)
## read counts along the chromosome
plot(log2(cpm_chr21[,2]+1),col='blue',pch=16,cex=0.5,xlab='window index',ylab='log2 signal',ylim=c(0,11
points(log2(cpm_chr21[,1]+1),col='red',pch=16,cex=0.5)
legend('topleft',c('ChIP','Input'),pch=16,col=c('red','blue'),bty='n')
```



It is noted that there are other procedures available for exploratory data analysis. We don't cover them here for the sake of time.

Peak detection

We will use R package **normr** (Helmuth et al. 2016) for peak calling. Function **enrichR** will be utilized to perform peak detection. This function will count the reads for 250bp windows, and compare ChIP and INPUT by fitting a mixture of binomial distributions. Then, function **summary** will show the general model fitting results, including the percentage of enriched windows and the q-values.

```
## peak detection based on 250bp windows
#peakfit = enrichR(treatment = chip,control=input,genome='hg38',verbose=FALSE) ## bug due to genome ver
summary(peakfit)
```

```
## NormRFit-class object
##
## Type:                   'enrichR'
## Number of Regions:      12353090
## Number of Components:   2
## Theta* (naive bg):      0.539
## Background component B: 1
##
## +++ Results of fit +++
## Mixture Proportions:
## Background       Class 1
##      89.5%         10.5%
## Theta:
## Background       Class 1
##      0.445         0.958
##
## Bayesian Information Criterion:  196856
##
## +++ Results of binomial test +++
## T-Filter threshold: 8
## Number of Regions filtered out: 12352369
## Significantly different from background B based on q-values:
## TOTAL:
##          ***      **       *       .            n.s.
## Bins       0     660      34      27       0       0
## %        0.0    23.6    24.8    25.8    25.8     0.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '  ' 1 'n.s.'
```

We can see that the total number of significant enriched peaks/windows is 694, spreading across different levels of q-values. One of the major benefits of using Bioconductor packages is that we can extract all info/results from one package and easily load them into other packages to facilitate data analysis. Here, we will export results from **normr** to GRanges format (package **GenomicRanges**), which is a generic data container for genomic data analysis.

```
## formalizing peak info with GRanges container
peaks = getRanges(peakfit)
peaks$sig = getEnrichment(peakfit) ## add enrichmend signal
peaks$pvalue = getPvalues(peakfit) ## add enrichment significance
peaks
```

```
## GRanges object with 12353090 ranges and 3 metadata columns:
```

```
##              seqnames            ranges strand | component          sig
##                 <Rle>         <IRanges>  <Rle> | <integer>    <numeric>
##          [1]     chr1             1-250      * |      <NA> -8.27017e-18
##          [2]     chr1           251-500      * |      <NA> -8.27017e-18
##          [3]     chr1           501-750      * |      <NA> -8.27017e-18
##          [4]     chr1          751-1000      * |      <NA> -8.27017e-18
##          [5]     chr1         1001-1250      * |      <NA> -8.27017e-18
##          ...      ...               ...    ... .       ...          ...
##   [12353086]     chrY 57226251-57226500      * |      <NA> -8.27017e-18
##   [12353087]     chrY 57226501-57226750      * |      <NA> -8.27017e-18
##   [12353088]     chrY 57226751-57227000      * |      <NA> -8.27017e-18
##   [12353089]     chrY 57227001-57227250      * |      <NA> -8.27017e-18
##   [12353090]     chrY 57227251-57227415      * |      <NA> -8.27017e-18
##                pvalue
##             <numeric>
##          [1]         1
##          [2]         1
##          [3]         1
##          [4]         1
##          [5]         1
##          ...       ...
##   [12353086]         1
##   [12353087]         1
##   [12353088]         1
##   [12353089]         1
##   [12353090]         1
##   -------
##   seqinfo: 24 sequences from an unspecified genome
```

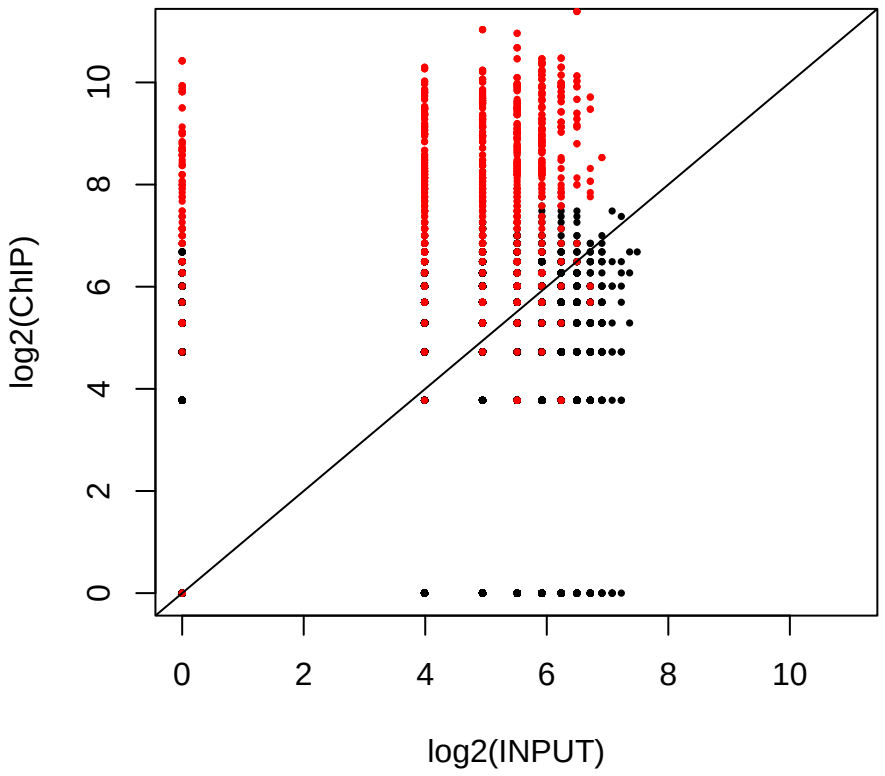
We can see most of the peaks/windows hold p-values as NA. These regions are actually not significantly enriched based on function **enrichR**. However, all regions are reported no matter the statuses of their significance. Thus, we should further filter these peaks using p-values.

```
## only consider a peak as significantly enriched if its q-value is less than 0.01
peakssig = peaks[which(peaks$pvalue<0.01)]
peakssig = peakssig[order(peakssig$pvalue)]
peakssig
```

```
## GRanges object with 706 ranges and 3 metadata columns:
##         seqnames            ranges strand | component       sig      pvalue
##            <Rle>         <IRanges>  <Rle> | <integer> <numeric>   <numeric>
##     [1]    chr21 25801501-25801750      * |         1   1.24785 9.73874e-35
##     [2]    chr21 25801751-25802000      * |         1   1.18900 3.17057e-28
##     [3]    chr21 43789251-43789500      * |         1   1.17497 7.37378e-27
##     [4]    chr21 43939251-43939500      * |         1   2.86317 9.60544e-27
##     [5]    chr21 43975501-43975750      * |         1   2.85912 2.15827e-26
##     ...      ...               ...    ... .       ...       ...         ...
##   [702]    chr21 37582501-37582750      * |         1  0.683697  0.00837267
##   [703]    chr21 42310001-42310250      * |         1  0.683697  0.00837267
##   [704]    chr21 44298001-44298250      * |         1  0.683697  0.00837267
##   [705]    chr21 45308751-45309000      * |         1  0.683697  0.00837267
##   [706]    chr21 45477751-45478000      * |         1  0.683697  0.00837267
##   -------
##   seqinfo: 24 sequences from an unspecified genome
```

After filtering based on p-values, we have resulted 706 enriched peaks/windows. Now we can go back to re-visit the previous figure and color it by enrichment significance.

```
## determine those 1000bp windows that are overlapped with the significant peaks
sigidx = queryHits(findOverlaps(window_gr,peakssig))
## different colors to separate significance (red) from non-significance
plot(log2(cpm_chr21[-sigidx,2]+1),log2(cpm_chr21[-sigidx,1]+1),pch=16,cex=0.5,xlab='log2(INPUT)',ylab='
points(log2(cpm_chr21[sigidx,2]+1),log2(cpm_chr21[sigidx,1]+1),pch=16,cex=0.5,col='red')
abline(a=0,b=1)
```

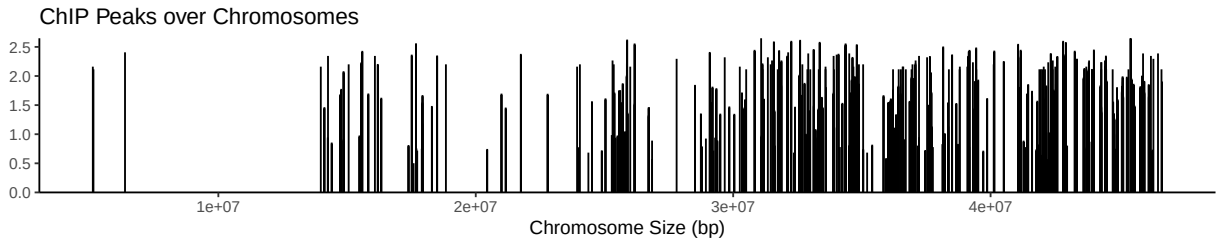


Peak annotation

Once peaks are identified, it is important and common to understand the characteristics of the peaks. We will use package **ChIPseeker** to annotate the significant peaks. Several key points will be covered here: (1) where do the significant peaks locate (2) how significant are the peak candidates (3) how does binding profile look like surrounding a given gene.

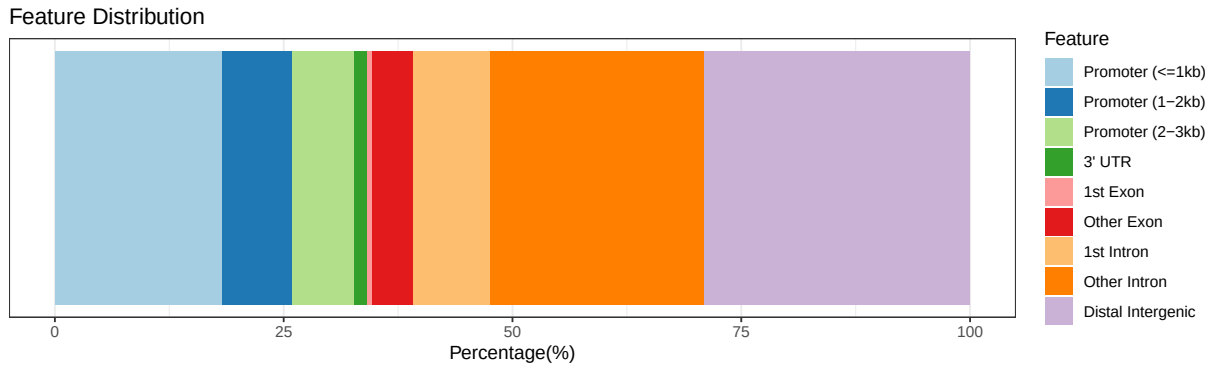
First, we can visualize the peaks along the chromosome based on barplot of peak enrichment scores. Some peaks show higher bars/significance while others lower. It is noted that genomic locations without bars do not necessarily mean there are no sequencing reads. It only suggests that reads are not significantly enriched.

```
## peaks along the chromosome
covplot(peakssig,chrs='chr21',lower=0.4,weightCol = 'sig')
```

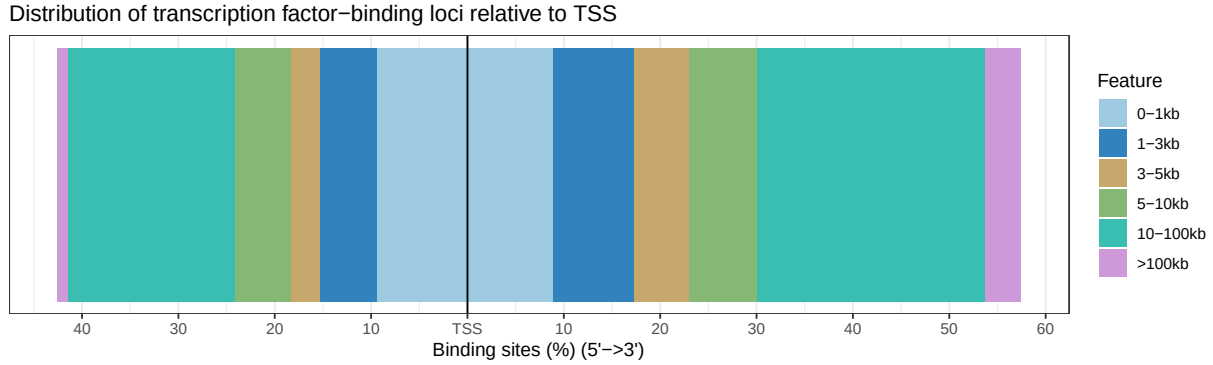


Another common task in ChIP-seq data analysis is to check how peaks are distributed in terms of their relations with genes. Here, we extract information of all annotated genes using Bioconductor package **TxDb.Hsapiens.UCSC.hg19.knownGene**. Function **annotatePeak** will then automatically calculate peak distribution based on these genes. We show following in two ways that how peaks are distributed across intergenic regions, promoters, introns and exons etc.

```
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
## annotate peaks
peakAnno <- annotatePeak(peakssig, tssRegion=c(-3000, 3000),
                         TxDb=txdb, annoDb="org.Hs.eg.db")
## visualize peak distribution
plotAnnoBar(peakAnno)
```




```
plotDistToTSS(peakAnno,
              title="Distribution of transcription factor-binding loci relative to TSS")
```

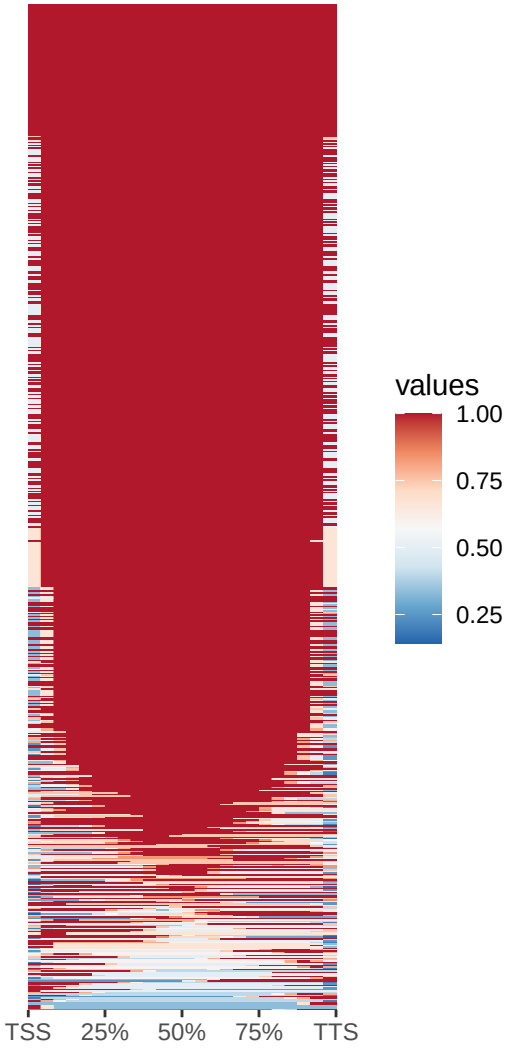


As we mentioned in our previous lecture, we can also visualize the signal intensities close to all ChIP-seq peaks. We do that by pile up all peak regions together, align them by their peak centers, extend all peaks to the same width, and visualize signal intensities using heatmap. See following code.

```
## choose a proper color you like for heatmap plot
cols = brewer.pal(9,'Greys')[(-7):(-9)]
## entend peaks into the same width, and align their centers
peakssigre = shift(resize(peakssig,6000),-round((6000-width(peakssig))/2))
## count read signals for every base pair in all extended peaks
tagMatrix <- getTagMatrix(peakssigre, windows=makeBioRegionFromGranges(peakssigre,by='gene',type='body'
```

```
## >> binning method is used...2023-08-17 5:58:56 PM
## >> preparing body regions by gene... 2023-08-17 5:58:56 PM
## >> preparing tag matrix by binning...  2023-08-17 5:58:56 PM
## >> preparing matrix for body region with no flank extension... 2023-08-17 5:58:56 PM
## >> 0 peaks(0%), having lengths smaller than 1000bp, are filtered... 2023-08-17 5:58:57 PM
```

```
## plot heatmap
tagHeatmap(tagMatrix)
```



Finally, we can visualize ChIP-seq signals around individual genes. To do so, we first extract the genomic coordinates of the genes of interest, *e.g.* ABG1. We will extend the gene region to include its promoter regions since promoter regions tend to be signal-enriched by transcription factors and are thus worthy to be checked in ChIP-seq studies. Here, we set promoter regions as the upstream of the transcription starting sites with 3000bp length. We extract ChIP-seq signals for the extended gene region and visualize the signals in a comparative manner between ChIP and INPUT.

```
## extract gene information
## you are encouraged to change gene names
genes = genes(txdb)
genes = genes[seqnames(genes)=='chr21']
symbl = unlist(lookUp(genes$gene_id, 'org.Hs.eg', 'SYMBOL'))
genes$gene_name = symbl[match(genes$gene_id,names(symbl))]
genes
```

```
## GRanges object with 444 ranges and 2 metadata columns:
##             seqnames            ranges strand |     gene_id   gene_name
```

```
##                <Rle>         <IRanges>  <Rle> | <character> <character>
##   100126693    chr21 43322417-43332039      - |   100126693   LINC00322
##   100129027    chr21 45827961-45837987      - |   100129027   PCBP3-AS1
##   100131902    chr21 30289145-30289514      - |   100131902   KRTAP25-1
##   100132288    chr21   9068325-9129773      - |   100132288     TEKT4P2
##   100133286    chr21 36069642-36126640      - |   100133286    CBR1-AS1
##         ...      ...               ...    ... .         ...         ...
##        9619    chr21 42199689-42297244      + |        9619       ABCG1
##        9875    chr21 32311018-32393012      - |        9875        URB1
##        9946    chr21 33589341-33643926      - |        9946      CRYZL1
##        9980    chr21 36156782-36294274      + |        9980       DOP1B
##        9992    chr21 34364006-34371381      + |        9992       KCNE2
##   -------
##   seqinfo: 711 sequences (1 circular) from hg38 genome
```

```
pickgenename = 'ABCG1'
genepick = genes[which(genes$gene_name==pickgenename)[1]]
## extend gene region to include promoters
genepick = reduce(c(promoters(genepick,upstream = 3000,downstream = 0),genepick))
## extract ChIP-seq coverage from both ChIP and INPUT
#cov_chip = coverage(resize(GRanges(readGAlignments(chip)),300))
#cov_input = coverage(resize(GRanges(readGAlignments(input)),300))
cov_chip_gene = cov_chip$chr21[start(genepick):end(genepick)]
cov_input_gene = cov_input$chr21[start(genepick):end(genepick)]
## plot coverage and label the transcription starting site
plot(cov_chip_gene,type = 'l',col='red',ylab='coverage',xlab='genomic location',xaxt = 'n',main=pickgene
lines(cov_input_gene,col='blue')
axis(1, at=round(seq(1,width(genepick),len=5)), labels=round(seq(start(genepick),end(genepick),len=5)))
abline(v=ifelse(strand(genepick)=='+',3000,width(genepick)-3000),col='green',lwd=2)
legend('top',c('ChIP','Input'),lty=1,col=c('red','blue'),bty='n')
```