The chromstaR user's guide

Aaron Taudt*

June 10, 2016

Contents

*aaron.taudt@gmail.com

1 Introduction

ChIP-seq has become the standard technique for assessing the genome-wide chromatin state of DNA. *chromstaR* provides functions for the joint analysis of multiple ChIP-seq samples. It allows peak calling for transcription factor binding and histone modifications with a narrow (e.g. H3K4me3, H3K27ac, ...) or broad (e.g. H3K36me3, H3K27me3, ...) profile. All analysis can be performed on each sample individually (=univariate), or in a joint analysis considering all samples simultaneously (=multivariate).

2 Outline of workflow

Every analysis with the *chromstaR* package starts from aligned reads in either BAM or BED format. In the first step, the genome is partitioned into non-overlapping, equally sized bins and the reads that fall into each bin are counted. These read counts serve as the basis for both the univariate and the multivariate peak- and broad-region calling. Univariate peak calling is done by fitting a three-state Hidden Markov Model to the binned read counts. Multivariate peak calling for $\mathcal{S}$ samples is done by fitting a $2^{\mathcal{S}}$-state Hidden Markov Model to all binned read counts.

3 Univariate analysis

3.1 Task 1: Peak calling for a narrow histone modification

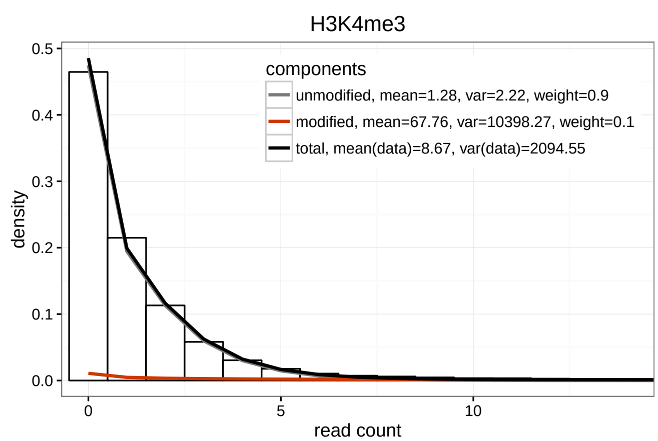
Examples of histone modifications with a narrow profile are H3K4me3, H3K9ac and H3K27ac in most human tissues. For such peak-like modifications, the bin size should be set to a value between 200bp and 1000bp.

```
library(chromstaR)
```

```
## === Step 1: Binning ===
# Get an example BED file
bedfile <- system.file("extdata","euratrans","lv-H3K4me3-BN-male-bio2-tech1.bed.gz",
                       package="chromstaRData")
# Get the chromosome lengths (see ?GenomeInfoDb::fetchExtendedChromInfoFromUCSC)
# This is only necessary for BED files. BAM files are handled automatically.
data(rn4_chrominfo)
head(rn4_chrominfo)

##   UCSC_seqlevel UCSC_seqlength NCBI_seqlevel
## 1          chrM          16300            MT
## 2         chr12       46782294            12
## 3         chr20       55268282            20
## 4         chr19       59218465            19
## 5         chr18       87265094            18
## 6         chr11       87759784            11

# We use bin size 1000bp and chromosome 12 to keep the example quick
binned.data <- binReads(bedfile, assembly=rn4_chrominfo, binsizes=1000,
                        chromosomes='chr12')
print(binned.data)

## GRanges object with 46782 ranges and 1 metadata column:
##           seqnames               ranges strand |    counts
##              <Rle>            <IRanges>  <Rle> | <integer>
##       [1]    chr12         [   1, 1000]      * |         0
##       [2]    chr12         [1001, 2000]      * |         0
##       [3]    chr12         [2001, 3000]      * |         0
##       [4]    chr12         [3001, 4000]      * |         0
##       [5]    chr12         [4001, 5000]      * |         0
##       ...      ...                  ...    ... .       ...
##   [46778]    chr12 [46777001, 46778000]      * |         2
##   [46779]    chr12 [46778001, 46779000]      * |         1
##   [46780]    chr12 [46779001, 46780000]      * |         0
##   [46781]    chr12 [46780001, 46781000]      * |         1
##   [46782]    chr12 [46781001, 46782000]      * |         1
##   -------
##   seqinfo: 1 sequence from an unspecified genome
```

```
## === Step 2: Peak calling ===
# We restrict the peak calling to 60 seconds to keep this example quick.
model <- callPeaksUnivariate(binned.data, max.time=60, verbosity=0)

## Replaced read counts > 500 by 500 in 111 bins.  Set option 'read.cutoff=FALSE' to disable this filtering.  This filtering was done to increase the speed of the HMM.

## Calculating states from posteriors ...

##  0.1s

## Making segmentation ...

##  0.11s
```

```
## === Step 3: Checking the fit ===
# For a narrow modification, the fit should look something like this,
# with the 'modified'-component near the bottom of the figure
plotHistogram(model) + ggtitle('H3K4me3')
```




```
## === Step 4: Export to genome browser ===
# We can export peak calls and binned read counts with
exportUnivariates(list(model), filename=tempfile(), what=c('peaks','counts'))
```

3.2 Task 2: Peak calling for a broad histone modification

Examples of histone modifications with a broad profile are H3K9me3, H3K27me3, H3K36me3, H4K20me1 in most human tissues. These modifications usually cover broad domains of the genome, and the enrichment is best captured with a bin size between 500bp and 2000bp.

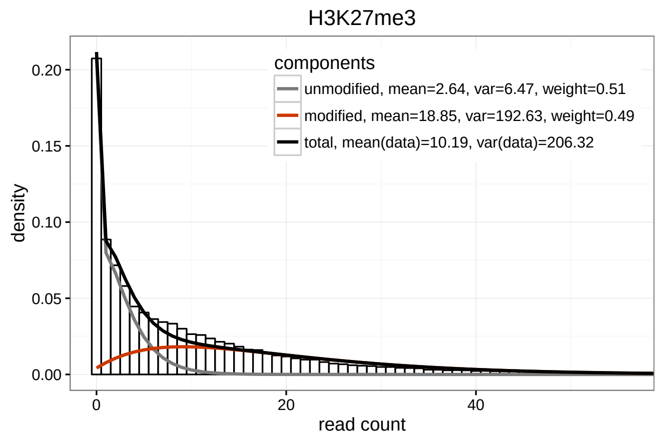
```
library(chromstaR)
```

```
## === Step 1: Binning ===
# Get an example BED file
bedfile <- system.file("extdata","euratrans","lv-H3K27me3-BN-male-bio2-tech1.bed.gz",
                       package="chromstaRData")
# Get the chromosome lengths (see ?GenomeInfoDb::fetchExtendedChromInfoFromUCSC)
# This is only necessary for BED files. BAM files are handled automatically.
data(rn4_chrominfo)
head(rn4_chrominfo)
# We use bin size 1000bp and chromosome 12 to keep the example quick
binned.data <- binReads(bedfile, assembly=rn4_chrominfo, binsizes=1000,
                       chromosomes='chr12')
```

```
## === Step 2: Peak calling ===
# We restrict the peak calling to 60 seconds to keep this example quick.
model <- callPeaksUnivariate(binned.data, max.time=60, verbosity=0)

## Calculating states from posteriors ...

##  0.08s

## Making segmentation ...

##  0.06s
```

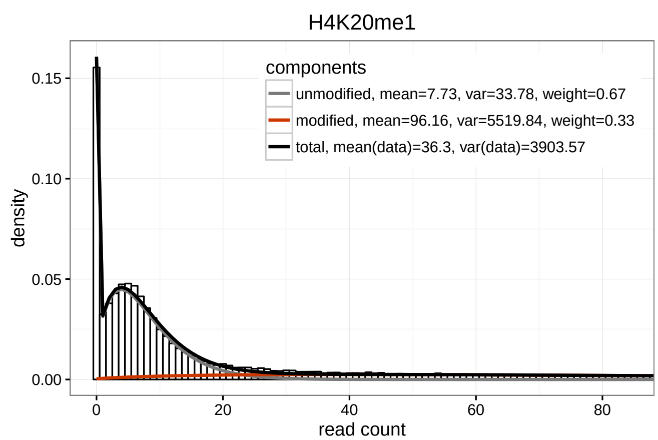
```
## === Step 3: Checking the fit ===
# For a broad modification, the fit should look something like this,
# with a 'modified'-component that fits the thick tail of the distribution.
```

```
plotHistogram(model) + ggtitle('H3K27me3')
```




```
## === Step 4: Export to genome browser ===
# We can export peak calls and binned read counts with
exportUnivariates(list(model), filename=tempfile(), what=c('peaks','counts'))
```

```
## === Step 1-3: Another example for mark H4K20me1 ===
bedfile <- system.file("extdata","euratrans","lv-H4K20me1-BN-male-bio1-tech1.bed.gz",
                       package="chromstaRData")
data(rn4_chrominfo)
binned.data <- binReads(bedfile, assembly=rn4_chrominfo, binsizes=1000,
                        chromosomes='chr12')
model <- callPeaksUnivariate(binned.data, max.time=60, verbosity=0)
plotHistogram(model) + ggtitle('H4K20me1')
```



3.3 Task 3: Peak calling for ATAC-seq, DNase-seq, FAIRE-seq, ...

Peak calling for ATAC-seq and DNase-seq is similar to the peak calling of a narrow histone modification (see section 3.1). FAIRE-seq experiments seem to exhibit a broad profile with our model, so the procedure is similar to the domain calling of a broad histone modification (see section 3.2).

4 Multivariate analysis

4.1 Task 1: Integrating multiple replicates

chromstaR can be used to call peaks with multiple replicates, without the need of prior merging. The underlying statistical model integrates information from all replicates to identify common peaks. It is, however, important to note that replicates with poor quality can affect the joint peak calling negatively. It is therefore recommended to first check the replicate quality and discard poor-quality replicates. The necessary steps for peak calling for an example ChIP-seq experiment with 4 replicates are detailed below.

```
library(chromstaR)
```

```
## === Step 1: Binning ===
# Let's get some example data with 3 replicates in spontaneous hypertensive rat (SHR)
file.path <- system.file("extdata","euratrans", package='chromstaRData')
bedfiles.good <- list.files(file.path, pattern="H3K27me3.*SHR", full.names=TRUE)[1:3]
# We fake a replicate with poor quality by taking a different mark entirely
bedfiles.poor <- list.files(file.path, pattern="H4K20me1.*SHR", full.names=TRUE)[1]
bedfiles <- c(bedfiles.good, bedfiles.poor)
# Obtain chromosome lengths. This is only necessary for BED files. BAM files are
# handled automatically.
data(rn4_chrominfo)
head(rn4_chrominfo)

##   UCSC_seqlevel UCSC_seqlength NCBI_seqlevel
## 1          chrM          16300            MT
## 2         chr12       46782294            12
## 3         chr20       55268282            20
## 4         chr19       59218465            19
## 5         chr18       87265094            18
## 6         chr11       87759784            11

# Define experiment structure
exp <- data.frame(file=bedfiles, mark='H3K27me3', condition='SHR', replicate=1:4,
                  pairedEndReads=FALSE)
# We use bin size 1000bp and chromosome 12 to keep the example quick
binned.data <- list()
for (bedfile in bedfiles) {
  binned.data[[basename(bedfile)]] <- binReads(bedfile, binsize=1000,
                                       assembly=rn4_chrominfo, chromosomes='chr12',
                                       experiment.table=exp)
}
```

```
## === Step 2: Univariate peak calling ===
# The univariate fit is obtained for each replicate
models <- list()
for (i1 in 1:length(binned.data)) {
  models[[i1]] <- callPeaksUnivariate(binned.data[[i1]], max.time=60)
}
```

```
## === Step 3: Check replicate correlation ===
# We run a multivariate peak calling on all 4 replicates
# A warning is issued because replicate 4 is very different from the others
multi.model <- callPeaksReplicates(models, max.time=60, eps=1)

## HMM: number of states = 16
## HMM: number of bins = 46782
## HMM: maximum number of iterations = none
## HMM: maximum running time = 60 sec
## HMM: epsilon = 1
## HMM: number of experiments = 4
##  Iteration              log(P)              dlog(P)     Time in sec
##          0                -inf                    -               0
## HMM: Precomputing densities ...
##  Iteration              log(P)              dlog(P)     Time in sec
##          0                -inf                    -               0
##          1      -548054.506310                  inf               0
##          2      -543443.215457          4611.290853               1
##          3      -543342.279125           100.936332               1
##          4      -543322.108413            20.170712               1
##          5      -543316.358595             5.749818               1
##          6      -543314.331591             2.027004               1
##          7      -543313.506624             0.824968               1
## HMM: Convergence reached!
## HMM: Recoding posteriors ...

## Warning in callPeaksReplicates(models, max.time = 60, eps = 1):  Your replicates cluster in 2 groups.  Consider redoing your analysis
with only the group with the highest average coverage:
## H3K27me3-SHR-rep4
## Replicates from groups with lower coverage are:
## H3K27me3-SHR-rep1
## H3K27me3-SHR-rep2
```

```
## H3K27me3-SHR-rep3

# Checking the correlation confirms that Rep4 is very different from the others
print(multi.model$replicateInfo$correlation)
```

```
## === Step 4: Peak calling with replicates ===
# We redo the previous step without the "bad" replicate
# Also, we force all replicates to agree in their peak calls
multi.model <- callPeaksReplicates(models[1:3], force.equal=TRUE, max.time=60)
```

```
## === Step 5: Export to genome browser ===
# Finally, we can export the results as BED file
exportMultivariate(multi.model, filename=tempfile(), what=c('peaks','counts'))
```

4.2 Task 2: Detecting differentially modified regions

chromstaR is extremely powerful in detecting differentially modified regions in two or more samples. The following example illustrates this on ChIP-seq data for H4K20me1 in brown norway (BN) and spontaneous hypertensive rat (SHR) in left-ventricle (lv) heart tissue. The mode of analysis is called *condition*, because all conditions are analyzed simultaneously.

```
library(chromstaR)
```

```
#=== Step 1: Preparation ===
## Prepare the file paths. Exchange this with your input and output directories.
inputfolder <- system.file("extdata","euratrans", package="chromstaRData")
outputfolder <- file.path(tempdir(), 'H4K20me1-example')

## Define experiment structure
data(experiment_table_H4K20me1)
print(experiment_table_H4K20me1)

##                                    file     mark condition replicate pairedEndReads
## 1  lv-H4K20me1-BN-male-bio1-tech1.bed.gz H4K20me1        BN         1          FALSE
## 2  lv-H4K20me1-BN-male-bio2-tech1.bed.gz H4K20me1        BN         2          FALSE
## 3     lv-input-BN-male-bio1-tech1.bed.gz    input        BN         1          FALSE
## 4     lv-input-BN-male-bio1-tech2.bed.gz    input        BN         2          FALSE
## 5 lv-H4K20me1-SHR-male-bio1-tech1.bed.gz H4K20me1       SHR         1          FALSE
## 6    lv-input-SHR-male-bio1-tech1.bed.gz    input       SHR         1          FALSE

## Define assembly
# This is only necessary if you have BED files, BAM files are handled automatically.
# For common assemblies you can also specify them as 'hg19' for example.
data(rn4_chrominfo)
head(rn4_chrominfo)

##   UCSC_seqlevel UCSC_seqlength NCBI_seqlevel
## 1          chrM          16300            MT
## 2         chr12       46782294            12
## 3         chr20       55268282            20
## 4         chr19       59218465            19
## 5         chr18       87265094            18
## 6         chr11       87759784            11
```

```
#=== Step 2: Run Chromstar ===
## Run ChromstaR
Chromstar(inputfolder, experiment.table=experiment_table_H4K20me1,
          outputfolder=outputfolder, numCPU=2, binsize=1000, assembly=rn4_chrominfo,
          prefit.on.chr='chr12', mode='condition')
```

```
## Results are stored in 'outputfolder' and can be loaded for further processing
list.files(outputfolder)

##  [1] "binned"               "BROWSERFILES"         "chrominfo.tsv"        "chromstaR.config"     "combined"
##  [6] "experiment_table.tsv" "multivariate"         "PLOTS"                "README.txt"           "replicates"
## [11] "univariate"

model <- get(load(file.path(outputfolder,'multivariate',
                            'multivariate_mode-condition_mark-H4K20me1.RData')))
```

```
## === Step 3: Construct differential and common states ===
diff.states <- stateBrewer(experiment_table_H4K20me1, mode='condition',
                           differential.states=TRUE)
```

```
print(diff.states)

##   combination state
## 1       [SHR]     1
## 2        [BN]     6

common.states <- stateBrewer(experiment_table_H4K20me1, mode='condition',
                             common.states=TRUE)
print(common.states)

##   combination state
## 1          []     0
## 2    [BN+SHR]     7
```

```
## === Step 5: Export to genome browser ===
# Export only differential states
exportMultivariate(model, filename=tempfile(),
                   what=c('peaks','counts','combinations'),
                   include.states=diff.states)
```

4.3 Task 3: Finding combinatorial chromatin states

Most experimental studies that probe several histone modifications are interested in combinatorial chromatin states. An example of a simple combinatorial state would be [H3K4me3+H3K27me3], which is also frequently called "bivalent promoter", due to the simultaneous occurrence of the promoter marking H3K4me3 and the repressive H3K27me3. Finding combinatorial states with *chromstaR* is equivalent to a multivariate peak calling. The following code chunks demonstrate how to find bivalent promoters and do some simple analysis. The mode of analysis is called *mark*, because all marks are analyzed simultaneously.

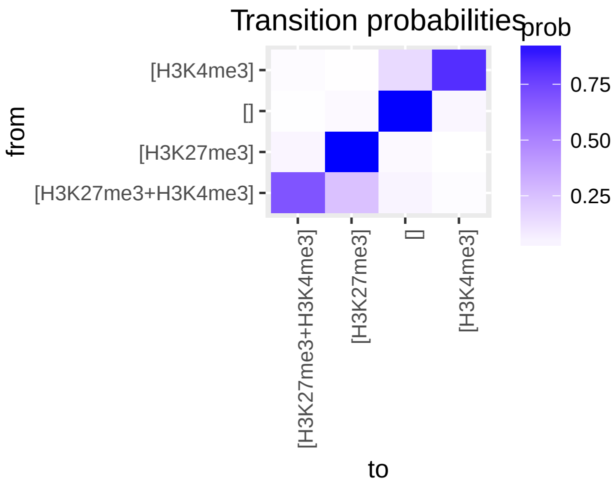
```
library(chromstaR)
```

```
#=== Step 1: Preparation ===
## Prepare the file paths. Exchange this with your input and output directories.
inputfolder <- system.file("extdata","euratrans", package="chromstaRData")
outputfolder <- file.path(tempdir(), 'SHR-example')

## Define experiment structure (SHR = spontaneous hypertensive rat)
data(experiment_table_SHR)
print(experiment_table_SHR)

##                                       file     mark condition replicate pairedEndReads
## 1 lv-H3K27me3-SHR-male-bio2-tech1.bed.gz H3K27me3       SHR         1          FALSE
## 2 lv-H3K27me3-SHR-male-bio2-tech2.bed.gz H3K27me3       SHR         2          FALSE
## 3  lv-H3K4me3-SHR-male-bio2-tech1.bed.gz  H3K4me3       SHR         1          FALSE
## 4  lv-H3K4me3-SHR-male-bio3-tech1.bed.gz  H3K4me3       SHR         2          FALSE
## 5    lv-input-SHR-male-bio1-tech1.bed.gz    input       SHR         1          FALSE

## Define assembly
# This is only necessary if you have BED files, BAM files are handled automatically.
# For common assemblies you can also specify them as 'hg19' for example.
data(rn4_chrominfo)
head(rn4_chrominfo)

##   UCSC_seqlevel UCSC_seqlength NCBI_seqlevel
## 1          chrM          16300            MT
## 2         chr12       46782294            12
## 3         chr20       55268282            20
## 4         chr19       59218465            19
## 5         chr18       87265094            18
## 6         chr11       87759784            11
```

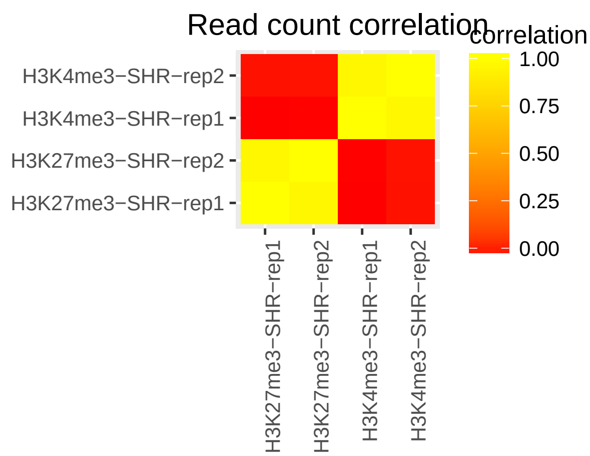
```
#=== Step 2: Run Chromstar ===
## Run ChromstaR
Chromstar(inputfolder, experiment.table=experiment_table_SHR,
          outputfolder=outputfolder, numCPU=2, binsize=1000, assembly=rn4_chrominfo,
          prefit.on.chr='chr12', mode='mark')
```

```
## Results are stored in 'outputfolder' and can be loaded for further processing
list.files(outputfolder)

##  [1] "binned"               "BROWSERFILES"         "chrominfo.tsv"        "chromstaR.config"     "combined"
##  [6] "experiment_table.tsv" "multivariate"         "PLOTS"                "README.txt"           "replicates"
## [11] "univariate"

model <- get(load(file.path(outputfolder,'multivariate',
                            'multivariate_mode-mark_condition-SHR.RData')))
# Obtain genomic frequencies for combinatorial states
genomicFrequencies(model)
```

```
##
##                []          [H3K4me3]         [H3K27me3] [H3K27me3+H3K4me3]
##        0.42995169         0.09356163         0.41840879         0.05807789

# Plot transition probabilities and read count correlation
plotTransitionProbs(model) + ggtitle('Transition probabilities')
```



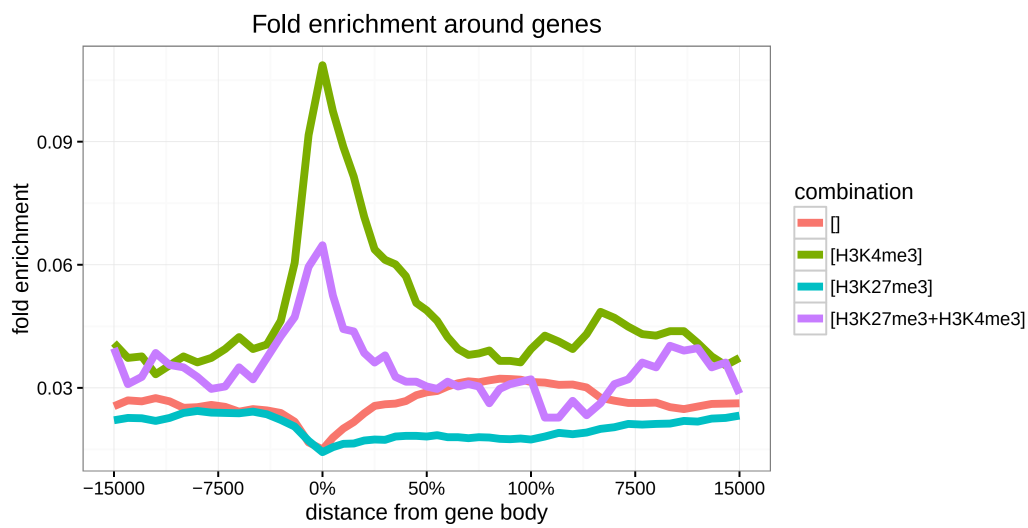

```
heatmapCountCorrelation(model) + ggtitle('Read count correlation')
```



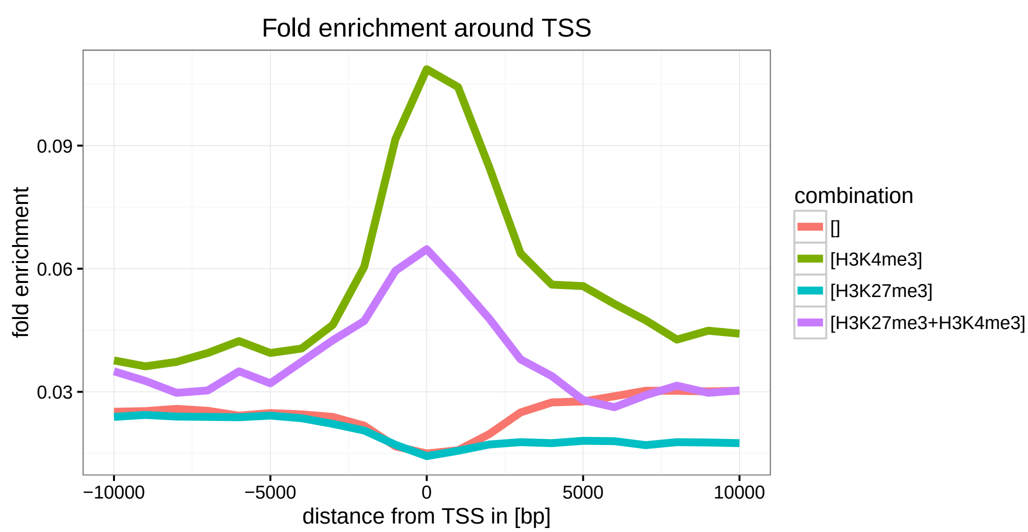

```
## === Step 3: Enrichment analysis ===
# Annotations can easily be obtained with the biomaRt package. Of course, you can
# also load them from file if you already have annotation files available.
library(biomaRt)
ensembl <- useMart('ENSEMBL_MART_ENSEMBL', host='may2012.archive.ensembl.org',
                   dataset='rnorvegicus_gene_ensembl')
genes <- getBM(attributes=c('ensembl_gene_id', 'chromosome_name', 'start_position',
                            'end_position', 'strand', 'external_gene_id',
                            'gene_biotype'),
               mart=ensembl)
# Transform to GRanges for easier handling
genes <- GRanges(seqnames=paste0('chr',genes$chromosome_name),
                 ranges=IRanges(start=genes$start, end=genes$end),
                 strand=genes$strand,
                 name=genes$external_gene_id, biotype=genes$gene_biotype)
print(genes)

## GRanges object with 29516 ranges and 2 metadata columns:
##           seqnames                 ranges strand |        name        biotype
##              <Rle>              <IRanges>  <Rle> | <character>    <character>
##       [1]    chr13     [1120899, 1121213]      - |   LOC682397 protein_coding
##       [2]    chr13     [1192186, 2293551]      - |   LOC304725 protein_coding
##       [3]    chr13     [3174383, 3175216]      + |                  pseudogene
##       [4]    chr13     [4377731, 4379174]      - |  D3ZPH4_RAT protein_coding
```

```
##        [5]     chr13     [4866302, 4866586]      - |  F1LZC7_RAT protein_coding
##        ...       ...                    ...    ... .         ...            ...
##    [29512]      chr6 [134310258, 134310338]      + |    SNORD113         snoRNA
##    [29513]      chr9 [  6920889,   6921049]      - |          U1          snRNA
##    [29514]     chr11 [ 40073746,  40073816]      - |    SNORD19B         snoRNA
##    [29515]      chr2 [233090372, 233090478]      - |          U6          snRNA
##    [29516]      chr6 [ 92917449,  92917541]      + |                      miRNA
##    -------
##    seqinfo: 22 sequences from an unspecified genome; no seqlengths
```

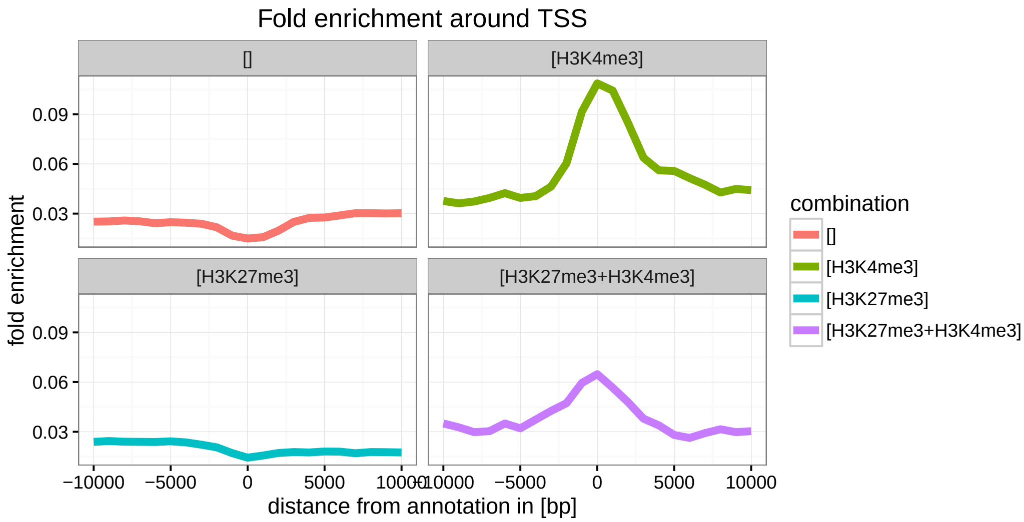
```r
# We expect promoter [H3K4me3] and bivalent-promoter signatures [H3K4me3+H3K27me3]
# to be enriched at transcription start sites.
plotFoldEnrichment(hmm = model, annotation = genes, bp.around.annotation = 15000) +
  ggtitle('Fold enrichment around genes') +
  xlab('distance from gene body')
```



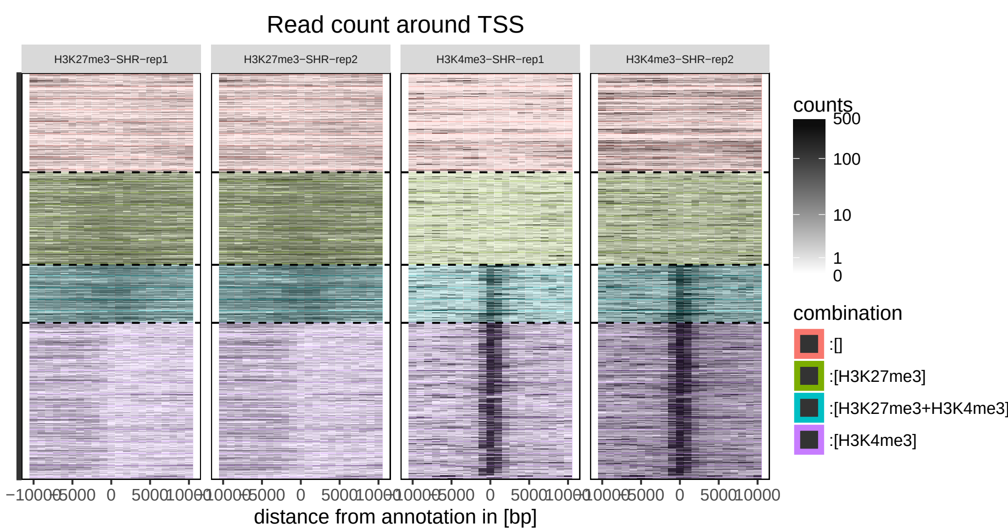

```r
# Plot enrichment only at TSS. We make use of the fact that TSS is the start of a gene.
plotFoldEnrichment(model, genes, region = 'start') +
  ggtitle('Fold enrichment around TSS') +
  xlab('distance from TSS in [bp]')
```



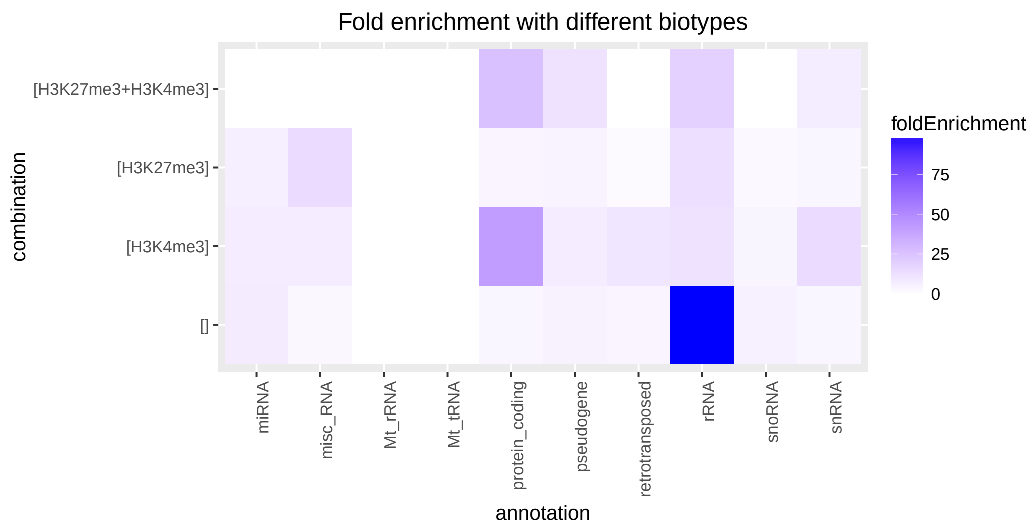

```r
# Note: If you want to facet the plot because you have many combinatorial states you
# can do that with
plotFoldEnrichment(model, genes, region = 'start') +
  facet_wrap(~ combination) + ggtitle('Fold enrichment around TSS')
```



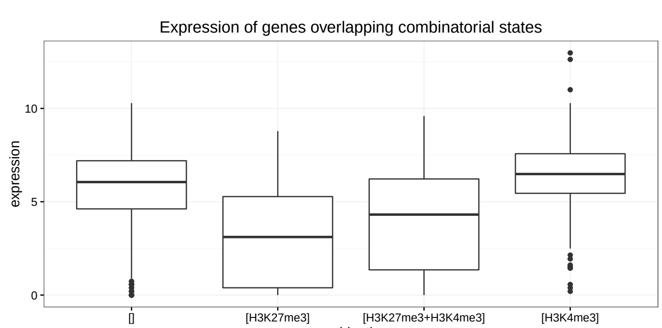

```
# Another form of visualization that shows every TSS in a heatmap
tss <- resize(genes, width = 3, fix = 'start')
plotEnrichCountHeatmap(model, tss) +
  theme(strip.text.x = element_text(size=6)) +
  ggtitle('Read count around TSS')
```



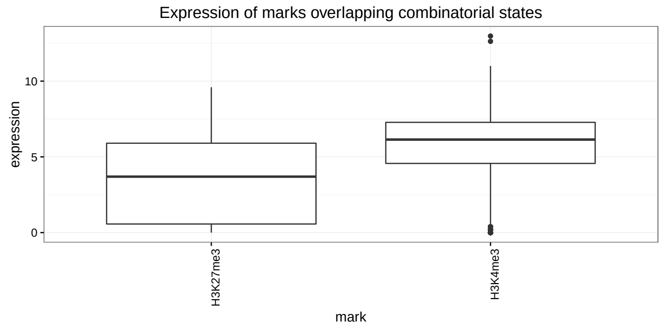

```
# Fold enrichment with different biotypes, showing that protein coding genes are
# enriched with (bivalent) promoter combinations [H3K4me3] and [H3K4me3+H3K27me3],
# while rRNA is enriched with the empty [] combination.
biotypes <- split(tss, tss$biotype)
plotFoldEnrichHeatmap(model, annotations=biotypes) + coord_flip() +
  ggtitle('Fold enrichment with different biotypes')
```




```
# === Step 4: Expression analysis ===
# Suppose you have expression data as well for your experiment. The following code
# shows you how to get the expression values for each combinatorial state.
data(expression_lv)
head(expression_lv)

##      ensembl_gene_id expression_BN expression_SHR
## 1 ENSRNOG00000000001           8.8            7.4
## 2 ENSRNOG00000000007          20.0           13.0
## 3 ENSRNOG00000000008           1.8            3.4
## 4 ENSRNOG00000000010           6.2          506.8
## 5 ENSRNOG00000000012          48.0           36.4
## 6 ENSRNOG00000000014          18.2           15.2

# We need to get coordinates for each of the genes
library(biomaRt)
ensembl <- useMart('ENSEMBL_MART_ENSEMBL', host='may2012.archive.ensembl.org',
                   dataset='rnorvegicus_gene_ensembl')
genes <- getBM(attributes=c('ensembl_gene_id', 'chromosome_name', 'start_position',
                            'end_position', 'strand', 'external_gene_id',
                            'gene_biotype'),
               mart=ensembl)
expr <- merge(genes, expression_lv, by='ensembl_gene_id')
# Transform to GRanges
expression.SHR <- GRanges(seqnames=paste0('chr',expr$chromosome_name),
                          ranges=IRanges(start=expr$start, end=expr$end),
                          strand=expr$strand, name=expr$external_gene_id,
                          biotype=expr$gene_biotype,
                          expression=expr$expression_SHR)
# We apply an asinh transformation to reduce the effect of outliers
expression.SHR$expression <- asinh(expression.SHR$expression)

## Plot
plotExpression(model, expression.SHR) +
  theme(axis.text.x=element_text(angle=0, hjust=0.5)) +
  ggtitle('Expression of genes overlapping combinatorial states')
```




```
plotExpression(model, expression.SHR, return.marks=TRUE) +
  ggtitle('Expression of marks overlapping combinatorial states')
```



4.4 Task 4: Finding differences between combinatorial chromatin states

Consider bivalent promoters defined by [H3K4me3+H3K27me3] at two different developmental stages, or in two different strains or tissues. This is an example where one is interested in *differences* between *combinatorial states*. The following example demonstrates how such an analysis can be done with *chromstaR*. We use a data set from the Euratrans project (downsampled to chr12) to find differences in bivalent promoters between brown norway (BN) and spontaneous hypertensive rat (SHR) in left-ventricle (lv) heart tissue.

`Chromstar` can be run in three different modes:

- *full*: Recommended mode if your (number of marks) * (number of conditions) is less or equal to 8. With 8 ChIP-seq experiments there are already $2^8 = 256$ combinatorial states which is the maximum that most computers can handle computationally for a human-sized genome at bin size 1000bp.
- *condition*: Choose this mode if you are interested in highly significant differences between conditions. The computational limit for the number of conditions is ~ 8 for a human-sized genome. Combinatorial states are not as accurate as in mode *mark* or *full*.
- **DEFAULT** *mark*: This mode will yield good combinatorial chromatin state calls for any number of marks and conditions. However, differences between conditions have more false positives than in mode *condition* or *full*.
- **separate**: (This mode can't be selected and is automatically produced. It is a simple concatenation of the univariate peak calls.)

```
library(chromstaR)
```

```
#=== Step 1: Preparation ===
## Prepare the file paths. Exchange this with your input and output directories.
inputfolder <- system.file("extdata","euratrans", package="chromstaRData")
outputfolder <- file.path(tempdir(), 'SHR-BN-example')

## Define experiment structure
data(experiment_table)
print(experiment_table)

##                                    file     mark condition replicate pairedEndReads
## 1   lv-H3K27me3-BN-male-bio2-tech1.bed.gz H3K27me3        BN         1          FALSE
## 2   lv-H3K27me3-BN-male-bio2-tech2.bed.gz H3K27me3        BN         2          FALSE
## 3  lv-H3K27me3-SHR-male-bio2-tech1.bed.gz H3K27me3       SHR         1          FALSE
## 4  lv-H3K27me3-SHR-male-bio2-tech2.bed.gz H3K27me3       SHR         2          FALSE
## 5  lv-H3K4me3-BN-female-bio1-tech1.bed.gz  H3K4me3        BN         1          FALSE
## 6    lv-H3K4me3-BN-male-bio2-tech1.bed.gz  H3K4me3        BN         2          FALSE
## 7   lv-H3K4me3-SHR-male-bio2-tech1.bed.gz  H3K4me3       SHR         1          FALSE
## 8   lv-H3K4me3-SHR-male-bio3-tech1.bed.gz  H3K4me3       SHR         2          FALSE
## 9      lv-input-BN-male-bio1-tech1.bed.gz    input        BN         1          FALSE
## 10     lv-input-BN-male-bio1-tech2.bed.gz    input        BN         2          FALSE
## 11    lv-input-SHR-male-bio1-tech1.bed.gz    input       SHR         1          FALSE

## Define assembly
# This is only necessary if you have BED files, BAM files are handled automatically.
# For common assemblies you can also specify them as 'hg19' for example.
data(rn4_chrominfo)
head(rn4_chrominfo)

##   UCSC_seqlevel UCSC_seqlength NCBI_seqlevel
## 1          chrM          16300            MT
## 2         chr12       46782294            12
## 3         chr20       55268282            20
```

```
## 4         chr19       59218465            19
## 5         chr18       87265094            18
## 6         chr11       87759784            11
```

```
#=== Step 2: Run Chromstar ===
## Run ChromstaR
Chromstar(inputfolder, experiment.table=experiment_table,
          outputfolder=outputfolder, numCPU=2, binsize=1000, assembly=rn4_chrominfo,
          prefit.on.chr='chr12', mode='condition')
```

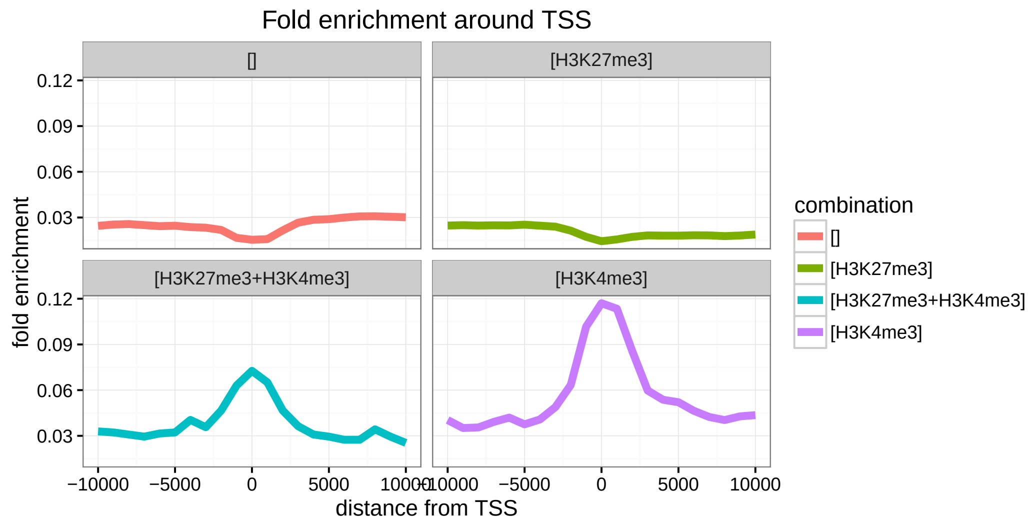
```
## Results are stored in 'outputfolder' and can be loaded for further processing
list.files(outputfolder)

##  [1] "binned"               "BROWSERFILES"         "chrominfo.tsv"        "chromstaR.config"     "combined"
##  [6] "experiment_table.tsv" "multivariate"         "PLOTS"                "README.txt"           "replicates"
## [11] "univariate"

model <- get(load(file.path(outputfolder,'combined',
                            'combined_mode-condition.RData')))
```

```
#=== Step 3: Analysis and export ===
## Obtain all genomic regions where the two tissues have different states
segments <- model$segments
diff.segments <- segments[segments$combination.SHR != segments$combination.BN]
# Let's be strict with the differences and get only those where both marks are different
diff.segments <- diff.segments[diff.segments$differential.score >= 1.9]
exportGRangesAsBedFile(diff.segments, trackname='differential_chromatin_states',
              filename=tempfile(), scorecol='differential.score')

## Warning in exportGRangesAsBedFile(diff.segments, trackname = "differential_chromatin_states", :  Column 'differential.score' should contain integer values between 0 and 1000 for compatibility with the UCSC convention.

## Obtain all genomic regions where we find a bivalent promoter in BN but not in SHR
bivalent.BN <- segments[segments$combination.BN == '[H3K27me3+H3K4me3]' &
                        segments$combination.SHR != '[H3K27me3+H3K4me3]']
## Obtain all genomic regions where BN and SHR have promoter signatures
promoter.BN <- segments[segments$transition.group == 'constant [H3K4me3]']

## Get transition frequencies
print(model$frequencies)

##        combination.BN    combination.SHR    frequency cumulative.frequency                          group
## 1                  []                 [] 4.397204e-01            0.4397204                zero transition
## 2          [H3K27me3]         [H3K27me3] 4.225771e-01            0.8622975            constant [H3K27me3]
## 3           [H3K4me3]          [H3K4me3] 8.368603e-02            0.9459835             constant [H3K4me3]
## 4  [H3K27me3+H3K4me3] [H3K27me3+H3K4me3] 4.912146e-02            0.9951050    constant [H3K27me3+H3K4me3]
## 5          [H3K27me3]                 [] 1.795562e-03            0.9969005      stage-specific [H3K27me3]
## 6                  []         [H3K27me3] 1.496302e-03            0.9983968      stage-specific [H3K27me3]
## 7                  []          [H3K4me3] 5.343936e-04            0.9989312       stage-specific [H3K4me3]
## 8          [H3K27me3] [H3K27me3+H3K4me3] 4.061391e-04            0.9993374                          other
## 9  [H3K27me3+H3K4me3]         [H3K27me3] 1.710059e-04            0.9995084                          other
## 10          [H3K4me3]                 [] 1.282545e-04            0.9996366       stage-specific [H3K4me3]
## 11          [H3K4me3] [H3K27me3+H3K4me3] 1.282545e-04            0.9997649                          other
## 12                 [] [H3K27me3+H3K4me3] 6.412723e-05            0.9998290 stage-specific [H3K27me3+H3K4me3]
## 13 [H3K27me3+H3K4me3]                 [] 6.412723e-05            0.9998931 stage-specific [H3K27me3+H3K4me3]
## 14 [H3K27me3+H3K4me3]          [H3K4me3] 6.412723e-05            0.9999572                          other
## 15         [H3K27me3]          [H3K4me3] 4.275149e-05            1.0000000                          other
```

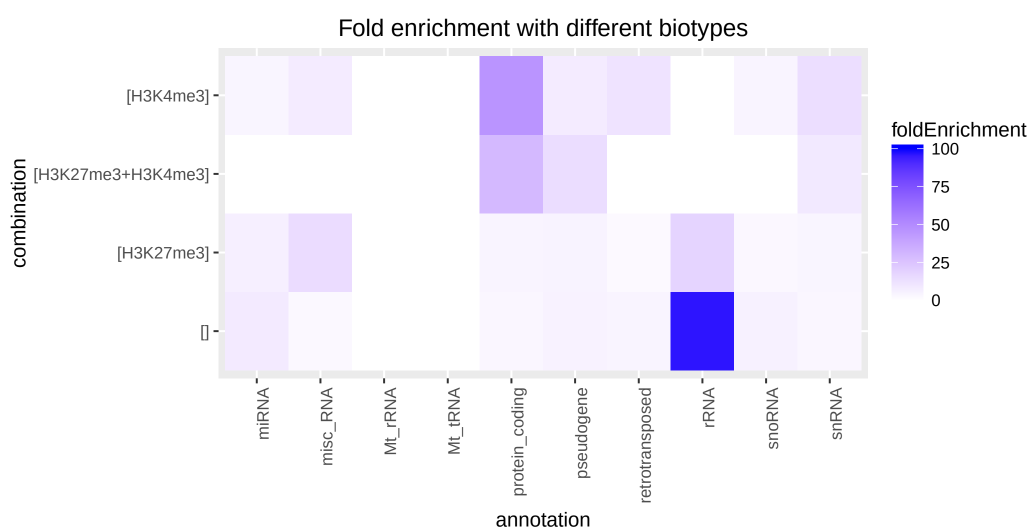
```
## === Step 4: Enrichment analysis ===
# Annotations can easily be obtained with the biomaRt package. Of course, you can
# also load them from file if you already have annotation files available.
library(biomaRt)
ensembl <- useMart('ENSEMBL_MART_ENSEMBL', host='may2012.archive.ensembl.org',
                   dataset='rnorvegicus_gene_ensembl')
genes <- getBM(attributes=c('ensembl_gene_id', 'chromosome_name', 'start_position',
                            'end_position', 'strand', 'external_gene_id',
                            'gene_biotype'),
               mart=ensembl)
# Transform to GRanges for easier handling
genes <- GRanges(seqnames=paste0('chr',genes$chromosome_name),
                 ranges=IRanges(start=genes$start, end=genes$end),
                 strand=genes$strand,
                 name=genes$external_gene_id, biotype=genes$gene_biotype)
print(genes)

## GRanges object with 29516 ranges and 2 metadata columns:
##           seqnames                 ranges strand |        name        biotype
##              <Rle>              <IRanges>  <Rle> | <character>    <character>
##       [1]    chr13     [1120899, 1121213]      - |   LOC682397 protein_coding
##       [2]    chr13     [1192186, 2293551]      - |   LOC304725 protein_coding
##       [3]    chr13     [3174383, 3175216]      + |                 pseudogene
##       [4]    chr13     [4377731, 4379174]      - |  D3ZPH4_RAT protein_coding
##       [5]    chr13     [4866302, 4866586]      - |  F1LZC7_RAT protein_coding
```

```
##        ...      ...                      ...    ... .          ...            ...
##    [29512]     chr6 [134310258, 134310338]      + |     SNORD113         snoRNA
##    [29513]     chr9 [  6920889,   6921049]      - |           U1          snRNA
##    [29514]    chr11 [ 40073746,  40073816]      - |     SNORD19B         snoRNA
##    [29515]     chr2 [233090372, 233090478]      - |           U6          snRNA
##    [29516]     chr6 [ 92917449,  92917541]      + |                        miRNA
##    -------
##    seqinfo: 22 sequences from an unspecified genome; no seqlengths
```

```
# We expect promoter [H3K4me3] and bivalent-promoter signatures [H3K4me3+H3K27me3]
# to be enriched at transcription start sites.
plots <- plotFoldEnrichment(hmm=model, annotation=genes, region='start')
plots[['BN']] + facet_wrap(~ combination) +
  ggtitle('Fold enrichment around TSS') +
  xlab('distance from TSS')
```




```
# Fold enrichment with different biotypes, showing that protein coding genes are
# enriched with (bivalent) promoter combinations [H3K4me3] and [H3K4me3+H3K27me3],
# while rRNA is enriched with the empty [] combination.
tss <- resize(genes, width = 3, fix = 'start')
biotypes <- split(tss, tss$biotype)
plots <- plotFoldEnrichHeatmap(model, annotations=biotypes)
plots[['BN']] + coord_flip() +
  scale_fill_gradient(low='white', high='blue', limits=c(0,100)) +
  ggtitle('Fold enrichment with different biotypes')
```



5 Output of function `Chromstar()`

`Chromstar()` is the workhorse of the *chromstaR* package and produces all the files that are necessary for downstream analyses. Here is an explanation of the *files* and **folders** you will find in your **outputfolder**:

- *chrominfo.tsv*:
  A tab-separated file with chromosome lengths.
- *chromstaR.config*:
  A text file with all the parameters that were used to run function `Chromstar()`.
- *experiment_table.tsv*:
  A tab-separated file of your experiment setup.
- **binned**:
  RData files with the results of the binnig step. Contains GRanges objects with binned genomic coordinates and read counts.
- **BROWSERFILES**:
  Bed files for upload to the UCSC genome browser. It contains files with combinatorial states ("_combinations.bed.gz"), underlying peak calls ("_peaks.bed.gz"), and read counts ("_counts.wig.gz"). !!Always check the "_peaks.bed.gz" files if you are satisfied with the peak calls. If not, there are ways to make the calls stricter (see section 6.1).
- →**combined**←:
  RData files with the combined results of the uni- and multivariate peak calling steps. This is what you want to use for downstream analyses. Contains RclasscombinedMultiHMM objects.
  - *combined_mode-separate.RData* Simple combination of univariate peak calls (replicates considered) without multivariate analysis.
  - *combined_mode-mark.RData* Combination of multivariate results for mode='mark'.
  - *combined_mode-condition.RData* Combination of multivariate results for mode='condition'.
  - *combined_mode-full.RData* Combination of multivariate results for mode='full'.
- **multivariate**:
  RData files with the results of the multivariate peak calling step. Contains RclassmultiHMM objects.
- **PLOTS**:
  Several plots that are produced by default. Please check the plots in subfolder **univariate-distributions** for irregularities (see section 3).
- **replicates**:
  RData files with the result of the replicate peak calling step. Contains multiHMM objects.
- **univariate**:
  RData files with the result of the univariate peak calling step. Contains RclassuniHMM objects.

6 FAQ

6.1 The peak calls are too lenient. Can I adjust the strictness of the peak calling?

The strictness of the peak calling can be controlled with a posterior cutoff. The Hidden Markov Model gives posterior probabilities for each peak, and based on these probabilites the model decides if a peak is present or not by picking the state with the highest probability. This way of peak calling leads to very lenient peak calls, and for some applications it may be desirable to obtain only very clear peaks. This can be achieved by setting a posterior cutoff. To follow the below example, please first run step 1 and 2 from section 4.4. !!Note that in general more peaks are obtained with a stricter cutoff, as this will lead to the split-up of previously broader peaks into several smaller, more well defined peaks.

```
model <- get(load(file.path(outputfolder,'combined',
                            'combined_mode-condition.RData')))
# Set a strict cutoff close to 1
model2 <- changePostCutoff(model, post.cutoff=0.999)
## Compare the number and width of peaks before and after cutoff adjustment
length(model$segments); mean(width(model$segments))

## [1] 4339
## [1] 10781.75
```

```
length(model2$segments); mean(width(model2$segments))

## [1] 6096
## [1] 7674.213
```

It is even possible to adjust the posterior cutoff differently for the different marks or conditions.

```
# Set a stricter cutoff for H3K4me3 than for H3K27me3
cutoffs <- c(0.8, 0.8, 0.8, 0.8, 0.999, 0.999, 0.999, 0.999)
names(cutoffs) <- model$info$ID
print(cutoffs)

##  H3K27me3-BN-rep1  H3K27me3-BN-rep2 H3K27me3-SHR-rep1 H3K27me3-SHR-rep2   H3K4me3-BN-rep1   H3K4me3-BN-rep2
##             0.800             0.800             0.800             0.800             0.999             0.999
##  H3K4me3-SHR-rep1  H3K4me3-SHR-rep2
##             0.999             0.999

model2 <- changePostCutoff(model, post.cutoff=cutoffs)
## Compare the number and width of peaks before and after cutoff adjustment
length(model$segments); mean(width(model$segments))

## [1] 4339
## [1] 10781.75

length(model2$segments); mean(width(model2$segments))

## [1] 4560
## [1] 10259.21
```

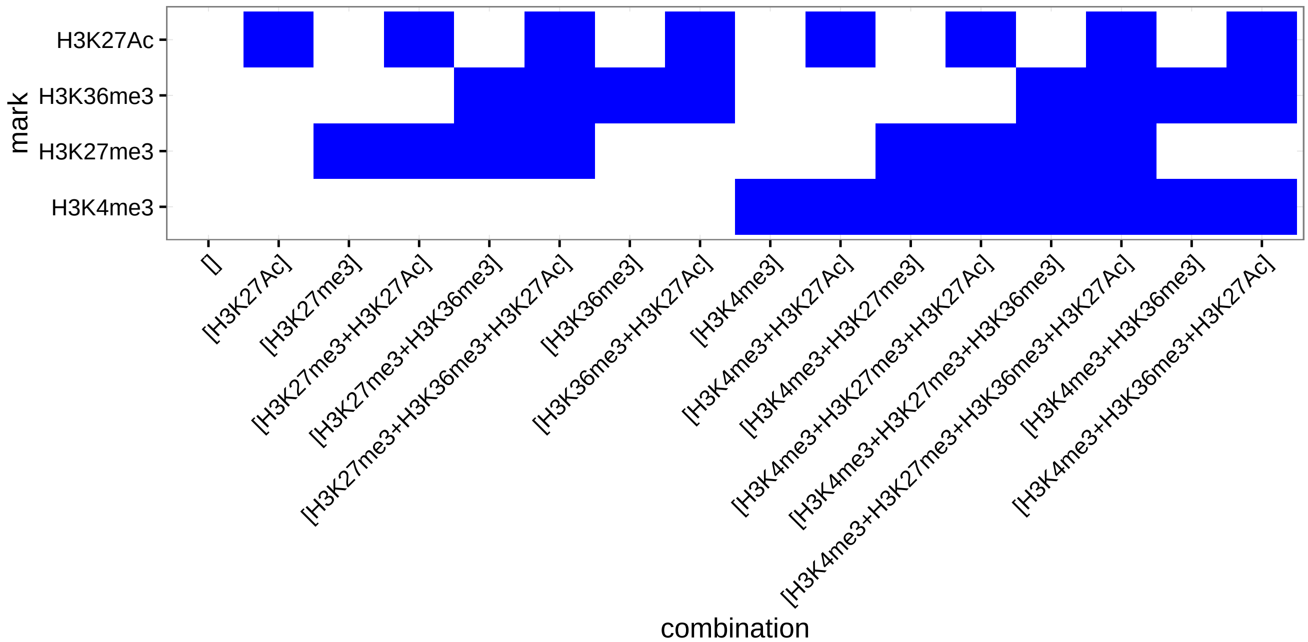
6.2 The combinatorial differences that chromstaR gives me are not convincing. Is there a way to restrict the results to a more convincing set?

You were interested in combinatorial state differences as in section 4.4 and checked the results in a genome browser. You found that some differences are convincing by eye and some are not. There are several possibilities to explore:

1. Run `Chromstar` in mode='condition' (instead of mode='mark') and see if the results improve.
2. You can play with the "differential.score" (see section 4.4, step 3) and export only differences with a high score. A differential score around 1 means that one modification is different, a score close to 2 means that two modifications are different etc. The score is calculated as the sum of differences in posterior probabilities between marks.
3. Check for bad replicates that are very different from the rest and exclude them prior to the analysis.

6.3 How do I plot a simple heatmap with the combinations?

```
heatmapCombinations(marks=c('H3K4me3', 'H3K27me3', 'H3K36me3', 'H3K27Ac'))
```



7 Session Info

```
sessionInfo()

## R version 3.3.0 (2016-05-03)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04 LTS
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=de_DE.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=de_DE.UTF-8    LC_MESSAGES=en_US.UTF-8    LC_PAPER=de_DE.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C             LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] biomaRt_2.28.0       chromstaR_0.99.2     chromstaRData_0.99.2 ggplot2_2.1.0        GenomicRanges_1.24.1
##  [6] GenomeInfoDb_1.8.1   IRanges_2.6.0        S4Vectors_0.10.1     BiocGenerics_0.18.0  knitr_1.13
## [11] Rcpp_0.12.5          devtools_1.11.1
##
## loaded via a namespace (and not attached):
##  [1] compiler_3.3.0             formatR_1.4                plyr_1.8.4                 highr_0.6
##  [5] XVector_0.12.0             bitops_1.0-6               iterators_1.0.8            tools_3.3.0
##  [9] zlibbioc_1.18.0            digest_0.6.9               RSQLite_1.0.0              evaluate_0.9
## [13] memoise_1.0.0              gtable_0.2.0               foreach_1.4.3              DBI_0.4-1
## [17] withr_1.0.1                stringr_1.0.0              Biostrings_2.40.2          grid_3.3.0
## [21] Biobase_2.32.0             AnnotationDbi_1.34.3       XML_3.98-1.4               BiocParallel_1.6.2
## [25] reshape2_1.4.1             magrittr_1.5               scales_0.4.0               Rsamtools_1.24.0
## [29] codetools_0.2-14           GenomicAlignments_1.8.1    SummarizedExperiment_1.2.2 BiocStyle_2.0.2
## [33] colorspace_1.2-6           labeling_0.3               stringi_1.1.1              RCurl_1.95-4.8
## [37] munsell_0.4.3              doParallel_1.0.10

warnings()

## NULL
```