histoneHMM (Version 1.6)

Matthias Heinig

March 8, 2016

1 Example workflow using data included in the package

Load the package

```
> library(histoneHMM)
```

Load the example data set

```
> data(rat.H3K27me3)
```

First we analyse both samples separately using the new high-level interface

```
> posterior = run.univariate.hmm("BN_H3K27me3.txt", data=BN, n.expr.bins=5,
+     em=TRUE, chrom="chr19", maxq=1-1e-3, redo=F)
```

This call also produces a number of files holding the parameter estimates and posterior probabilities. We can now load the fit from the file and inspect its quality visually.

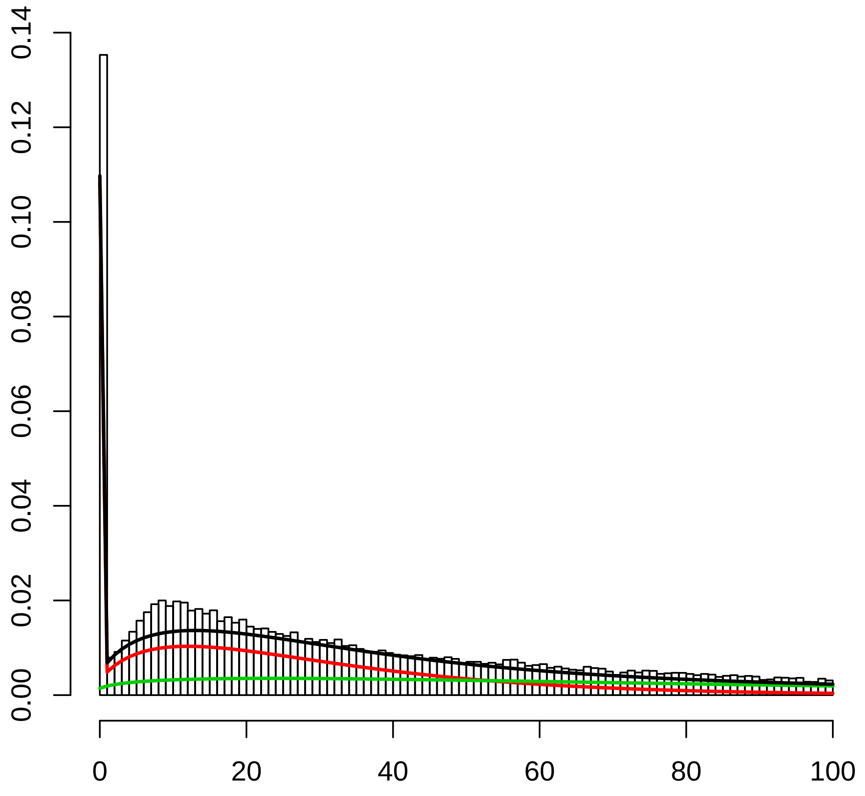
```
> n = load("BN_H3K27me3-zinba-params-em.RData")
> zinba.mix = get(n)
```

For the visualization we ignore the long tail of the distribution, otherwise it is difficult to see.

```
> max = 100
> x = BN[BN[,"chrom"] == "chr19","signal"]
> x[x > max] = NA
```

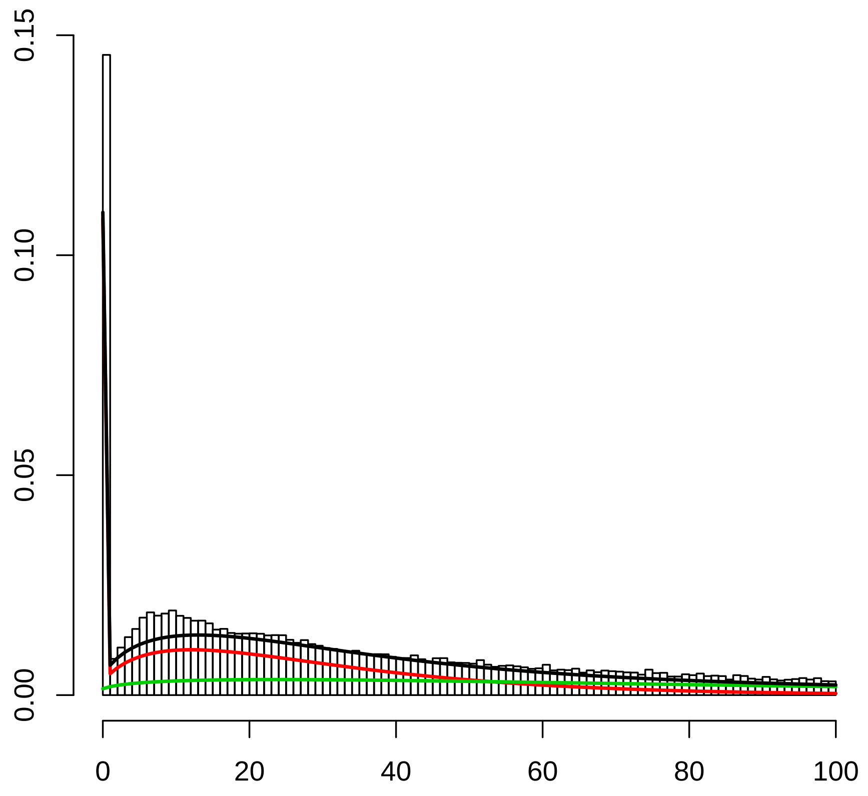
Visualize the fit

```
> plotDensity(model=zinba.mix, x=x, xlim=c(0, max), add=FALSE,
+             col="black", alpha=1)
```



Analysis of the second sample is carried out analogously.

```
> posterior = run.univariate.hmm("SHR_H3K27me3.txt", data=SHR, n.expr.bins=5,
+     em=TRUE, chrom="chr19", maxq=1-1e-3, redo=F)
> ## load the fit from the file
> n = load("BN_H3K27me3-zinba-params-em.RData")
> zinba.mix = get(n)
> ## for the visualization we ignore the long tail of the distribution
> max = 100
> x = SHR[SHR[,"chrom"] == "chr19","signal"]
> x[x > max] = NA
> ## visualize the fit
> plotDensity(model=zinba.mix, x=x, xlim=c(0, max), add=FALSE,
+             col="black", alpha=1)
```



After validating both fits we continue with the bivariate analysis

```
> bivariate.posterior = run.bivariate.hmm("BN_H3K27me3.txt", "SHR_H3K27me3.txt",
+     outdir="BN_vs_SHR_H3K27me3/", data1=BN, data2=SHR,
+     sample1="BN", sample2="SHR", n.expr.bins=5, maxq=1-1e-3,
+     em=TRUE, chrom="chr19")
> ## call regions
> BN.regions = callRegions(bivariate.posterior, 0.5, "BN", NULL)
> # GR2gff(BN.regions, "BN_vs_SHR_H3K27me3/BN_specific.gff")
>
> SHR.regions = callRegions(bivariate.posterior, 0.5, "SHR", NULL)
> # GR2gff(SHR.regions, "BN_vs_SHR_H3K27me3/SHR_specific.gff")
```

2 Preprocessing data from bam files

histoneHMM works on the read counts from equally sized bins over the genome. Optionally, gene expression data can be used to guide the parameter estimation. Here we show how the data is preprocessed. In order to get the full information we need a bam file containing the aligned ChIP-seq reads, a table of gene expression values, the lengths of the chromosomes and a gene annotation.

On our website we provide example data that can be downloaded for this example:

```
> system("wget http://histonehmm.molgen.mpg.de/data/expression.txt")
> system("wget http://histonehmm.molgen.mpg.de/data/BN.bam")
> system("wget http://histonehmm.molgen.mpg.de/data/BN.bam.bai")
> system("wget http://histonehmm.molgen.mpg.de/data/chroms.txt")
> system("wget http://histonehmm.molgen.mpg.de/data/ensembl59-genes.gff")
```

Load the expression data. In this experiment we have 5 replicates of each strain, so we take the mean expression values.

```
> expr = read.csv("expression.txt", sep="\t")
> mean.expr = apply(expr[,1:5], 1, mean)
```

The genome object that we pass to the function needs to provide a seqlengths function, we can use either the BSGenome packages from bioconductor, or simply create GRanges.

```
> chroms = read.table("chroms.txt", stringsAsFactors=FALSE)
> sl = chroms[,2]
> names(sl) = chroms[,1]
> genome = GRanges(seqlengths=sl)
```

Load the gene coordinates such that we can assign the expression values of a gene to all bins that overlap the gene.

```
> genes = gff2GR("ensembl59-genes.gff", "ID")
> data = preprocess.for.hmm("BN.bam", genes, bin.size=1000, genome,
+     expr=mean.expr, chr=c("chr19", "chr20"))
> write.table(data, "BN.txt", sep="\t", quote=F, row.names=F)
```

3 Command line interface

You will find two executable R scripts in the bin directory of the package. You can locate these scripts as follows:

```
R
system.file("bin/histoneHMM_call_regions.R",
            package="histoneHMM")
system.file("bin/histoneHMM_differential.R",
            package="histoneHMM")
```

Copy these files to a directory that is in your path to use them.

3.1 Calling of broad domains

This section describes the use of histoneHMM to analyze a single sample and call broad domains. When the analysis starts from the aligned reads, the minimal set of input parameters consists of:

- a bam file from a ChIP-seq experiment,
- a tab separated file with the name and length of each chromosome.

To perform a first quality and plausibility check histoneHMM can generate diagnostic plots to compare histone modification levels grouped by gene expression. Parameters for this analysis are:

- a file with gene annotation in gff format,
- a tab separated file with gene expression values.

The gene annotation should only contain gene bodies and contain ID='id' name-value pairs in field nine. The gene expression file should use the same ids (first column) and length normalized (and ideally log transformed) expression (second column and contain no header line.

In the following we will use a H3K27me3 DEEP data set from a HepG2 cell line ("01_HepG2_LiHG_Ct2") to demonstrate the use of the command line interface. Since we are also interested in the diagnostic plots we need the corresponding gene expression and annotation data. Here we use DEEP RNA-seq data and the gencode annotation.

Get gene expression counts using 'htseq-count' although it is not length normalized. Alternatively cufflinks or other tools can be used.

```
i="01_HepG2_LiHG_Ct1_tRNA_K.tophat2.20141113"
samtools sort -n ${i}.bam ${i}_name_sorted
samtools view ${i}_name_sorted.bam | htseq-count \
 --stranded=reverse - gencode.v19.annotation.gtf \
 > ${i}_counts.txt
```

Prepare the gene annotation such that it contains only gene bodies of protein coding genes, though other types of genes could also be included.

```
cat gencode.v19.annotation.gtf | \
 awk '{if ($3 == "gene") print $0}' | \
 grep "protein_coding" > gencode.v19.annotation-genes.gtf
```

Obtain the chromosome length information from UCSC.

```
wget http://hgdownload.cse.ucsc.edu/goldenpath/hg19/database/chromInfo.txt.gz
gunzip chromInfo.txt.gz
```

Finally we can run the histoneHMM analysis

```
histoneHMM_call_regions.R -c chromInfo.txt \
 -e ${i}_counts.txt -t chr20 \
 -o 01_HepG2_LiHG_Ct2_H3K27me3_histoneHMM
```

This will produce a number of output files all starting with the path prefix given in the '-o prefix' option:

suffix	file content
.txt	results of the preprocessing (counts)
-em-posterior.txt	the results of the region calling for each bin
-zinba-emfit.pdf	plot of the count histogram and the fit of the mixture distribution
-zinba-params-em.txt	the parameters of the fitted mixture distribution
-zinba-params-em.RData	the model as an object 'zinba.mix' in the RData file
-regions.gff	the region calls in gff format

3.2 Differential regions

To call differential regions between two samples for instance healthy and diseased samples, both samples first have to be processed individually as described above. Once both mixture fits have been verifyed we can run:

```
histoneHMM_call_differential.R \
 --outdir differential
 --sample1 HepG2 \
 --sample2 primary chromInfo.txt \
 01_HepG2_LiHG_Ct2_H3K27me3_histoneHMM.txt \
 01_HepG2_LiHG_Ct2_H3K27me3_histoneHMM.txt
```

This produce the following files in the directory specified in the –outdir option with prefix 'sample1'-vs-'sample2':

suffix	file content
.txt	the results of the region calling for each bin
-'sample1'.gff	regions only modified in sample1 in gff format
-'sample2'.gff	regions only modified in sample2 in gff format
-mod_both.gff	regions modified in both samples in gff format
-unmod_both.gff	regions unmodified in both samples in gff format
-zinbacopula-params.txt	parameters of the multivariate mixture