epigenomix — Epigenetic and gene expression data normalization and integration with mixture models

Hans-Ulrich Klein, Martin Schäfer

March 13, 2013

Contents

1 Introduction

This package provides methods for an integrative analysis of gene expression and epigenetic data, especially histone ChIP-seq data. Histone modifications are an epigenetic key mechanism to activate or repress the expression of genes. Several data sets consisting of matched microarray expression data and histone modification data measured by ChIP-seq have been published. However, both data types are often analysed separately and results are compared afterwards. The methods implemented here are designed to detect genes that are differentially expressed between two conditions due to an altered histone modification and are suitable for very small sample sizes.
Briefly, the following workflow is described in this documnet:

1. Matching of both data types by assigning the number of ChIP-seq reads aligning within the promoter region of a gene to the expression value of that gene

2. Normalization of ChIP-seq values

3. Calculation of a correlation score for each gene by multiplying the standardized difference of ChIP-seq values by the standardized difference of expression values

4. Fitting a (Bayesian) mixture model to this score: The implicit assignment of genes to mixture components is used to classify genes into one of the following groups: (i) Genes with equally directed differences in both data sets, (ii) genes with reversely directed differences in both data sets and (iii) genes with no differences in at least one of the two data sets. Group (iii) is represented by centred normal components whereas an exponential component is used for group (i) and a mirrored exponential component for group (ii).

2 Data preprocessing and normalization

2.1 Gene expression data

First, we load an example gene expression data set. The data set consists of four samples. Two wild type replicates and two *CEBPA* knock-out replicates. The differences between *CEBPA* knock-down and wild type samples are of interest. The data set is stored as an *ExpressionSet* object and was reduced to a few probesets on chromosome 1.

```
> library(epigenomix)
> data(eSet)
> pData(eSet)
```

```
           CEBPA
CEBPA_WT_a    wt
CEBPA_WT_b    wt
CEBPA_KO_a    ko
CEBPA_KO_b    ko
```

Data was measured using Affymetrix Mouse Gene 1.0 ST arrays and RMA normalized. See packages *affy* and *Biobase* how to process affymetrix gene expression data.

2.2 Histone ChIP-seq data

The example histone ChIP-seq data is stored as *GRangesList* object:

```
> data(mappedReads)
> names(mappedReads)
```

```
[1] "CEBPA_WT_1" "CEBPA_KO_1"
```

There are two elements within the list. One *CEBPA* wild type and one knock-out sample. Most of the originally obtained reads were removed to reduce storage space. Further, the reads were extended towards the 3 prime end to the mean DNA fragment size of 200bps and duplicated reads were removed. See R packages *Rsamtools* and *GenomicRanges* how to read in and process sequence reads.

2.3 Data matching

The presented ChIP-seq data measured H3K4me3 histone modifications. This modification primarily occures at promoter regions. Hence, we assign ChIP-seq values to probesets by counting the number of reads lying wihtin the promoter of the measured transcript. Therefore, we first create a list with one element for each probeset that stores the Ensemble transcript IDs of all transcripts measured by that probeset:

```
> probeToTrans <- fData(eSet)$transcript
> probeToTrans <- strsplit(probeToTrans, ",")
> names(probeToTrans) <- featureNames(eSet)
```

Next, we need the transcriptional start sites for each transcript.

```
> data(transToTSS)
> head(transToTSS)
```

```
     ensembl_transcript_id chromosome_name transcript_start
159     ENSMUST00000001172               1         36547201
441     ENSMUST00000003219               1         39535802
631     ENSMUST00000004829               1        171559193
766     ENSMUST00000006037               1         13374083
1202    ENSMUST00000013842               1        172206804
1306    ENSMUST00000015460               1        171767127
     strand
159      -1
441       1
631       1
766      -1
1202     -1
1306      1
```

Such a data frame can be obtained e.g. using *biomaRt*:

```
> library("biomaRt")
> transcripts <- unique(unlist(transToTSS))
> mart <- useMart("ensembl", dataset="mmusculus_gene_ensembl")
> transToTSS <- getBM(attributes=c("ensembl_transcript_id",
     "chromosome_name", "transcript_start",
     "transcript_end", "strand"),
     filters="ensembl_transcript_id",
     values=transcripts, mart=mart)
```

Having these information, the promoter region for each probeset can be calculated unsing `matchProbeToPromoter`. Argument `mode` defines how probesets with multiple transcripts should be handled.

```
> promoters <- matchProbeToPromoter(probeToTrans,
     transToTSS, promWidth=6000, mode="union")
> promoters[["10345616"]]
```

```
GRanges with 2 ranges and 1 metadata column:
      seqnames              ranges strand |       probe
         <Rle>           <IRanges>  <Rle> | <character>
  [1]        1 [37869206, 37875205]      + |    10345616
  [2]        1 [37887407, 37893406]      - |    10345616
```

```
  ---
  seqlengths:
    1
   NA
```

Note that some promoter regions, like for probeset `"10345616"`, may consist of more than one interval.
Finally, `summarizeReads` is used to count the number of reads within the promoter regions:

```
> chipSetRaw <- summarizeReads(mappedReads, promoters, summarize="add")
> chipSetRaw

ChIPseqSet (storageMode: lockedEnvironment)
assayData: 180 features, 2 samples
  element names: chipVals
protocolData: none
phenoData
  sampleNames: CEBPA_WT_1 CEBPA_KO_1
  varLabels: totalCount
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:

> head(chipVals(chipSetRaw))

         CEBPA_WT_1 CEBPA_KO_1
10344803        145        401
10344813        145        401
10344897          2          8
10345007          8          6
10345037         69        122
10345099         38         90
```

The method returns an object of class *ChIPseqSet*, which is derived from class *eSet* and is the ChIP-seq counterpart to *ExpressionSet*.

3 ChIP-seq data normalization

It may be necessary to normalize ChIP-seq data due to different experimental conditions during ChIP.

```
> chipSet <- normalizeChIP(chipSetRaw, method="quantile")
```

In addition to quantile normalization, other methods like the method presented by [Anders and Huber, 2010] are available.

```
> par(mfrow=c(1,2))
> plot(chipVals(chipSetRaw)[,1], chipVals(chipSetRaw)[,2],
      xlim=c(1,600), ylim=c(1,600), main="Raw")
> plot(chipVals(chipSet)[,1], chipVals(chipSet)[,2],
      xlim=c(1,600), ylim=c(1,600), main="Quantile")
```

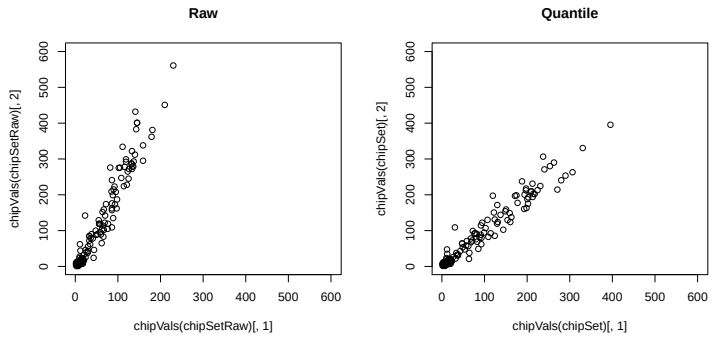

Figure 1: Raw and quantile normalized ChIP-seq data.

4 Data integration

In order to integrate both data types, a correlation score Z (motivated by the work of [Schäfer *et al.*, 2012]) can be calculated by multiplying the standardized difference of gene expression values with the standardized difference of ChIP-seq values. Prior to this, pheno type information must be added to the `chipSet` object.

```
> eSet$CEBPA

[1] wt wt ko ko
Levels: ko wt

> sampleNames(chipSet)

[1] "CEBPA_WT_1" "CEBPA_KO_1"

> chipSet$CEBPA <- factor(c("wt", "ko"))
> pData(chipSet)

           totalCount CEBPA
CEBPA_WT_1       8687    wt
CEBPA_KO_1      17122    ko

> intData <- integrateData(eSet, chipSet,
     factor="CEBPA", reference="wt")
> head(intData)

          expr_ko  expr_wt chipseq_ko chipseq_wt          z
10354832 8.864536 8.392561      193.0      202.5 -0.8048761
10359770 7.161367 7.305733      213.0      224.5  0.2980229
10355974 7.956849 7.850496      214.5      271.0 -1.0786664
10348378 5.384252 5.339577       49.0       85.5 -0.2927146
10353775 4.780612 4.700385       15.0       13.5  0.0216021
10352827 6.175612 5.873558        8.5        8.5  0.0000000
```

5 Classification by mixture models

5.1 Maximum likelihood approach

We now fit a mixture model to the correlation score Z. The model consists of two normal components with fixed $\mu = 0$. These two components should capture Z values close to zero, i.e. genes that show no differences between wild type and knock-out in at least one of the two data sets. The positive (negative) Z scores are represented by a (mirrored) exponential component. Parameters are estimated using the EM-algorithm as implemented in the method `mlMixModel`.

```
> mlmm = mlMixModel(intData[,"z"],
     normNull=c(2, 3), expNeg=1, expPos=4,
     sdNormNullInit=c(0.5, 1), rateExpNegInit=0.5, rateExpPosInit=0.5,
     pi=rep(1/4, 4))

> mlmm

MixModel object
    Number of data points:  180
    Number of components:   4
        1: ExpNeg
             rate = 1.532987
           weight pi = 0.2219707
           classified data points: 30
        2: NormNull
             mean = 0
             sd = 0.01644812
           weight pi = 0.2154126
           classified data points: 48
        3: NormNull
             mean = 0
             sd = 0.1213587
           weight pi = 0.3526906
           classified data points: 70
        4: ExpPos
             rate = 0.6931467
           weight pi = 0.2099261
           classified data points: 32
```

The method returns an object of class *MixModelML*, a subclass of *MixModel*. We now plot the model fit and the classification results:

```
> par(mfrow=c(1,2))
> plotComponents(mlmm, xlim=c(-2, 2), ylim=c(0, 3))
> plotClassification(mlmm)
```

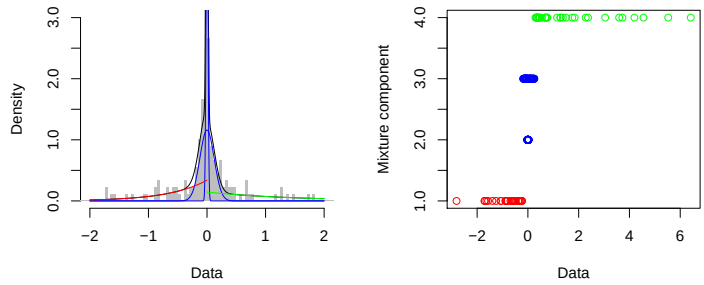

Figure 2: Model fit and classification results of the maximum likelihood approach.

5.2 Bayesian approach

Alternatively, an Bayesian approach can be used.

```
> set.seed(1515)
> bayesmm = bayesMixModel(intData[,"z"],
     normNull=c(2, 3), expNeg=1, expPos=4,
     sdNormNullInit=c(0.5, 1), rateExpNegInit=0.5, rateExpPosInit=0.5,
     shapeNorm0=c(10, 10), scaleNorm0=c(10, 10), shapeExpNeg0=0.01,
     scaleExpNeg0=0.01, shapeExpPos0=0.01, scaleExpPos0=0.01,
     pi=rep(1/4, 4), itb=2000, nmc=8000, thin=5)
```

`bayesMixModel` returns an object of class *MixModelBayes*, which is also a subclass of *MixModel*.

```
> bayesmm
```

```
MixModel object
    Number of data points:  180
    Number of components:   4
        1: ExpNeg
             rate = 0
           weight pi = 0.005949889
           classified data points: 0
        2: NormNull
             mean = 0
             sd = 0.0712299
```

```
   weight pi = 0.2435747
   classified data points: 96
3: NormNull
     mean = 0
     sd = 0.6347255
   weight pi = 0.4605196
   classified data points: 71
4: ExpPos
     rate = 0.1145572
   weight pi = 0.2899559
   classified data points: 13
```

The same methods for plotting the model fit and classification can be applied.

```
> par(mfrow=c(1,2))
> plotComponents(bayesmm, xlim=c(-2, 2), ylim=c(0, 3))
> plotClassification(bayesmm, method="mode")
```

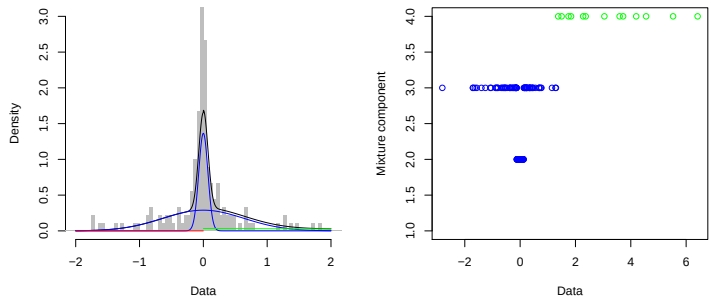

Figure 3: Model fit and classification results of the Bayesian approach.

Note, that the parameters 'burn in' (`itb`) and 'number of iterations' (`nmc`) have to be choosen carefully. The method `plotChains` should be used to assess the convergence of the markov chains for each parameter. The settings here lead to a short runtime, but are unsuitable for real applications.

Both models tend to classify more genes to the positive component (component 4) than to the negative one (component 1):

```
> table(classification(mlmm, method="maxDens"),
      classification(bayesmm, method="mode"))

    2  3  4
1   0 30  0
```

```
  2 48  0  0
  3 48 22  0
  4  0 19 13
```

This is in line with the fact, that H3K4me3 occurs in the promoters of active genes. Since each z corresponds to a probeset (and so to at least one transcript), the corresponding microarray annotation packages can be used to obtain e.g. the gene symbols of all positivly classified z scores.

```
> posProbes <- rownames(intData)[classification(bayesmm, method="mode") == 4]
> library("mogene10sttranscriptcluster.db")
> unlist(mget(posProbes, mogene10sttranscriptclusterSYMBOL))
```

References

[Anders and Huber, 2010] S. Anders and W. Huber (2010) Differential expression analysis for sequence count data. *Genome Biol.*, **11**(10), R106.

[Schäfer *et al.*, 2012] M. Schäfer, O. Lkhagvasuren, H.-U. Klein *et al.* (2012) Integrative analyses for Omics data: A Bayesian mixture model to assess the concordance of ChIP-chip and ChIP-seq measurements. *J Toxicol Environ Health A.*, **75**(8-10), 461–470.