iSeq: Bayesian Hierarchical Modeling of ChIP-seq Data Through Hidden Ising Models

Qianxing Mo

April 14, 2011

Department of Epidemiology and Biostatistics
Memorial Sloan-Kettering Cancer Center
moq@mskcc.org

Contents

1 Introduction **1**

2 The NRSF ChIP-seq Data **2**

3 Example — Analyze the NRSF Data **2**
3.1 Build dynamic signal profiles using function 'mergetag' 3
3.2 Model dynamic signal profiles using function 'iSeq1' 4
3.3 Call enriched regions detected by iSeq1 using function 'peakreg' 6
3.4 Model dynamic signal profiles using function 'iSeq2' 9
3.5 Call enriched regions detected by iSeq2 using function 'peakreg' 10

4 Tips **11**

5 Parallel Computation **14**

6 Citing iSeq **14**

1 Introduction

This package implements the models proposed by Mo (2010) for ChIP-seq data analysis, which are the extensions of the models proposed for ChIP-chip data (Mo and Liang, 2010a,b). The package can be used to analyze ChIP-seq data with or without controls.

This package processes ChIP-seq data in three steps.

- Build a dynamic signal profile for each chromosome. To create a dynamic signal profile, sequence tags are aggregated into non-overlapping dynamic bins whose lengths vary

according to local tag density (See the help file of function 'mergetag'). The number of tags falling in each bin is counted. A dynamic signal profile is made of the bin-based tag counts and the corresponding genomic positions on the chromosome. The bins and their tag counts are ordered along the chromosome according to their genomic positions.

- Model the dynamic signal profiles. It is to model the tag counts on the chromosome. The models assume that each bin is associated with a binary latent variable $X_i \in (1, -1)$, where i denotes the ID for the bin, and $X_i = 1$ denotes that the bin is an enriched bin, and -1 for a non-enriched bin. In the first stage, conditioning on the latent variable X_i, the bin-based tag counts are modeled by Poisson-Gamma distributions. If $X_i = -1$, the tag count for bin i is assumed to follow a Poisson distribution with parameter λ_0, where $\lambda_0 \sim \text{Gamma}(a_0, b_0)$. If $X_i = 1$, the tag count for bin i is assumed to follow a Poisson distribution with parameter λ_1, where $\lambda_1 \sim \text{Gamma}(a_1, b_1)$. $\text{Gamma}(a, b)$ denotes a gamma distribution with mean a/b and variance a/b^2. In the second stage, the latent variable is modeled by ferromagnetic Ising models. The Gibbs sampler and Metropolis algorithm are used to simulate from the posterior distributions of the model parameters.

- Call enriched regions. The posterior probabilities for the bins in the enriched state ($X_i = 1$) are used for statistical inference. A bin with a high posterior probability in the enriched state will provide strong evidence that the bin is enriched. Enriched bins are then merged into enriched regions.

For more information, we refer the user to Mo and Liang (2010 a, b) because the manuscript (Mo, 2010c) is still under review. The major difference for modeling ChIP-chip and ChIP-seq data is at the first stage, where normal distributions are used for ChIP-chip data, while Poisson distributions are used for ChIP-seq data.

2 The NRSF ChIP-seq Data

Johnson et al.(2007) carried out genome-wide identification of the binding sites of human neuron-restrictive silencer factor (NRSF) in Jurkat T cells. We use the data of chromosomes 22 and Y to illustrate the proposed method. The NRSF sequence tags (25 bp) were generated by the Illumina/Solexa sequencing platform, and mapped to The human genome May 2004 (hg17). We only use the uniquely mapped tags (up to two mismatches) for the analysis. Note that iSeq package doesn't provide functions to read the raw data generated by the sequencers. However, it should not be difficult to prepare the data in the format as shown in the following.

3 Example — Analyze the NRSF Data

First, let's load the library and check the data. There are three ChIP and control samples, respectively.

```
> library(iSeq)
> data(nrsf)
> names(nrsf)

[1] "chipFC1592" "chipFC1862" "chipFC2002" "mockFC1592" "mockFC1862"
[6] "mockFC2002"
```

3.1 Build dynamic signal profiles using function 'mergetag'

The following two steps just merge the ChIP and control samples, respectively.

```
> chip = rbind(nrsf$chipFC1592, nrsf$chipFC1862, nrsf$chipFC2002)
> mock = rbind(nrsf$mockFC1592, nrsf$mockFC1862, nrsf$mockFC2002)
> print(chip[1:3, ])

      chr position strand
38  chr22 48039379      F
104 chr22 28163725      R
180 chr22 38814016      R

> print(mock[1:3, ])

     chr position strand
15 chr22 35510328      F
57 chr22 25441949      F
90 chr22 31761090      F
```

We suggest building the signal profiles using dynamic bins (80, 40, 20, 10 bp) and 10 tags as the threshold for triggering bin size change for the NRSF data, which is equivalent to setting maxlen=80,minlen=10 and ntagcut=10 for the arguments of function 'megertag'. Of course, the user can use other settings according to the sequencing depth of the experiments. According to my experience, the dynamic profiles with bin sizes (80, 40, 20, 10 bp) work well for both deeply and not-deeply sequenced data. However, the users may need to adjust argument 'ntagcut' to reflect the sequencing depth. For example, for deeply sequenced data, the users may set ntagcut = 20, or 30, etc. In general, increasing the value of 'ntagcut' leads to bigger bin sizes used for building the dynamic profiles and less regions called enriched. Note, by default, the bin sizes decrease or increase by a factor of 2, thus the user should set maxlen = (2^n)*minlen, where n is an integer.

```
> tagct = mergetag(chip = chip, control = mock, maxlen = 80, minlen = 10,
+     ntagcut = 10)
> print(tagct[1:3, ])

    chr   gstart     gend adjct ipct1 ipct2 conct1 conct2
1 chr22 14433408 14433408     1     0     1      0      0
2 chr22 14436138 14436138     1     1     0      0      0
3 chr22 14436262 14436262     1     0     1      0      0
```

See the help file for function 'mergetag' for the meanings of the tagct columns.

The user can quickly get the 'enriched' regions without calculating statistical confidence using function 'peakreg'. For example, if we claim that a bin with adjusted tag count > 10 is an enriched bin, the 'enriched' regions can be obtained by using the following code. Let's use the chromosome 22 data as an example.

```
> tagct22 = tagct[tagct[, 1] == "chr22", ]
> reg0 = peakreg(chrpos = tagct22[, 1:3], count = (tagct22[, 5:6] -
+     tagct22[, 7:8]), pp = (tagct22[, 4] > 10), cutoff = 0, method = "ppcut",
+     maxgap = 200)
> print(dim(reg0))

[1] 77 11

> print(reg0[1:3, ])

    chr   gstart     gend rstart rend  peakpos meanpp ct1 ct2 ct12  sym
1 chr22 15913755 15913832    636  636 15913793   1.00  13  13   26 0.50
2 chr22 15975695 15976050    802  825 15975944   0.96 307 309  616 0.50
3 chr22 17330186 17330558   2568 2576 17330378   0.56 125 101  226 0.45
```

Note the second argument 'count = ' is just used for inferring the predicted exact binding sites (or peaks) of the enriched regions. Here, we use the adjusted tag counts without truncating at 0. The user can also just use the ChIP tag counts. Usually it will not make much difference. The last column (sym) in 'reg0' indicates whether the forward and reverse tag counts are symmetrical in the regions. It is to measure whether the regions are bimodal, which is a typical characteristic of binding regions. The values range from 0.5 (the perfect symmetry) to 0 (asymmetry). The user can order the regions based on 'sym' and 'ct12' to get the 'top' binding regions. For example,

```
> reg0 = reg0[order(reg0$sym, reg0$ct12, decreasing = TRUE), ]
> print(reg0[1:5, ])

     chr   gstart     gend rstart  rend  peakpos meanpp ct1 ct2 ct12 sym
61 chr22 43691646 43692284  45213 45243 43692138   0.90 937 950 1887 0.5
47 chr22 37544780 37545578  34350 34384 37545187   0.77 466 461  927 0.5
2  chr22 15975695 15976050    802   825 15975944   0.96 307 309  616 0.5
46 chr22 36183334 36184046  31831 31862 36183626   0.72 265 270  535 0.5
1  chr22 15913755 15913832    636   636 15913793   1.00  13  13   26 0.5
```

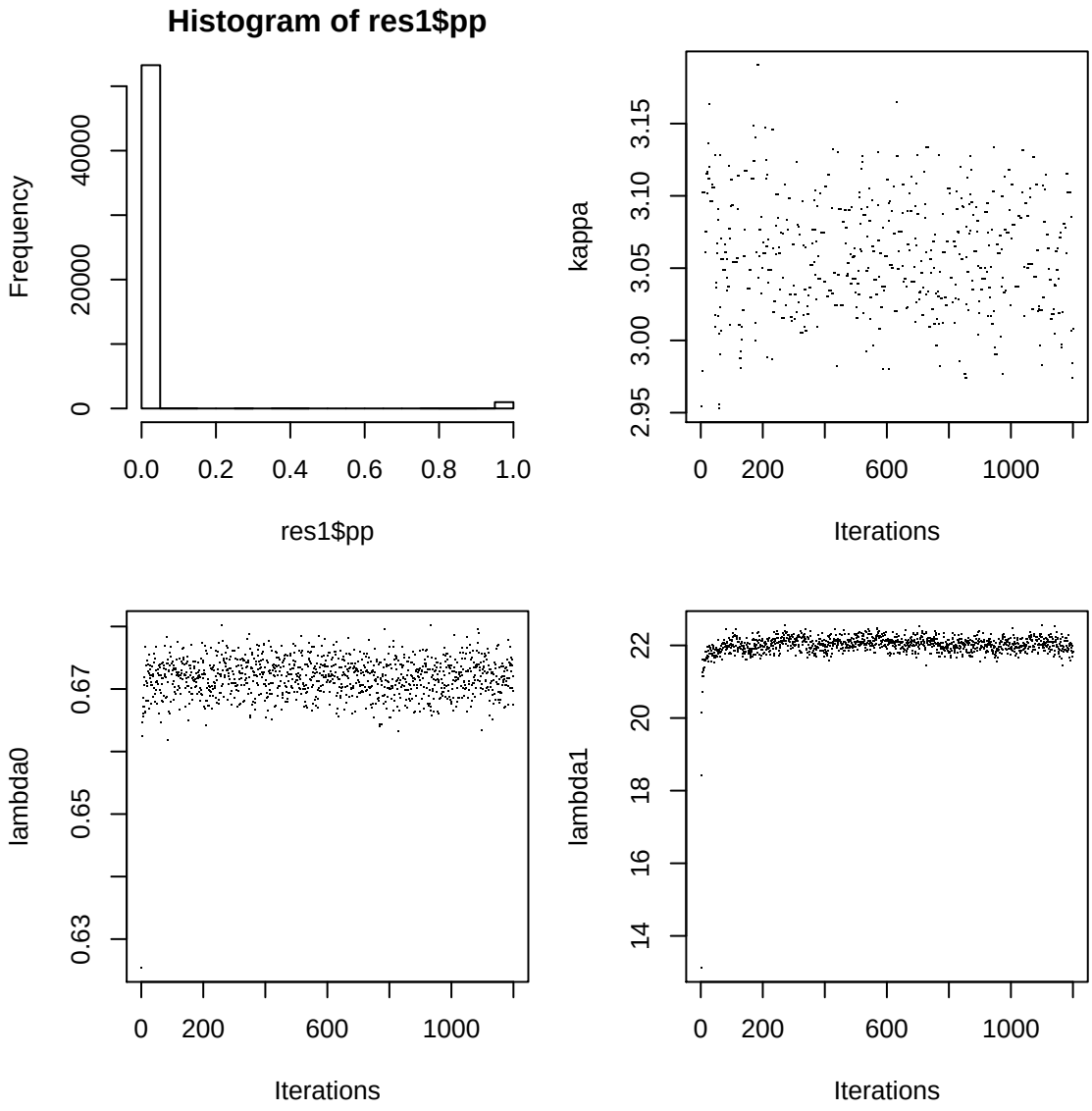
3.2 Model dynamic signal profiles using function 'iSeq1'

Function iSeq1 implements a fully Bayesian hidden Ising model in which the latent variable X is modeled by the standard 1D Ising model. The columns 1-4 of tagct are the signal profiles for modeling. Note that the column 4 contains the adjusted tag counts of the ChIP samples. For one sample analysis, it is the total tag counts for the forward and reverse chains.

```
> set.seed(777)
> res1 = iSeq1(Y = tagct22[, 1:4], gap = 200, burnin = 200, sampling = 1000,
+     ctcut = 0.95, a0 = 1, b0 = 1, a1 = 5, b1 = 1, k0 = 3, mink = 0,
+     maxk = 10, normsd = 0.1, verbose = FALSE)
```

Plot the model parameters to see whether they converge. In general, the MCMC chains have converged when the parameters fluctuate around the modes of their distributions. If there is an obvious trend(e.g. continuous increase or decrease), the user should increase the number of iterations in the burn-in and/or sampling phases. If the chains do not mix well, the user can adjust the argument normsd to see how it affects the results.

```
> par(mfrow = c(2, 2), mar = c(4.1, 4.1, 2, 1))
> hist(res1$pp)
> plot(res1$kappa, pch = ".", xlab = "Iterations", ylab = "kappa")
> plot(res1$lambda0, pch = ".", xlab = "Iterations", ylab = "lambda0")
> plot(res1$lambda1, pch = ".", xlab = "Iterations", ylab = "lambda1")
```



From the trace plots, we see the chains converge quite fast.

3.3 Call enriched regions detected by iSeq1 using function 'peakreg'

Call the enriched regions detected by iSeq1 using 0.5 posterior probability (pp) cutoff. Note the argument count is the net tag counts for the two sample analysis. The net tag counts are not truncated at zero, but this doesn't matter because it is just used for inferring the center of the enriched region, which is usually the true binding site. The user can also use the ChIP tag counts only. The results are little different. See the help file of peakreg for details.

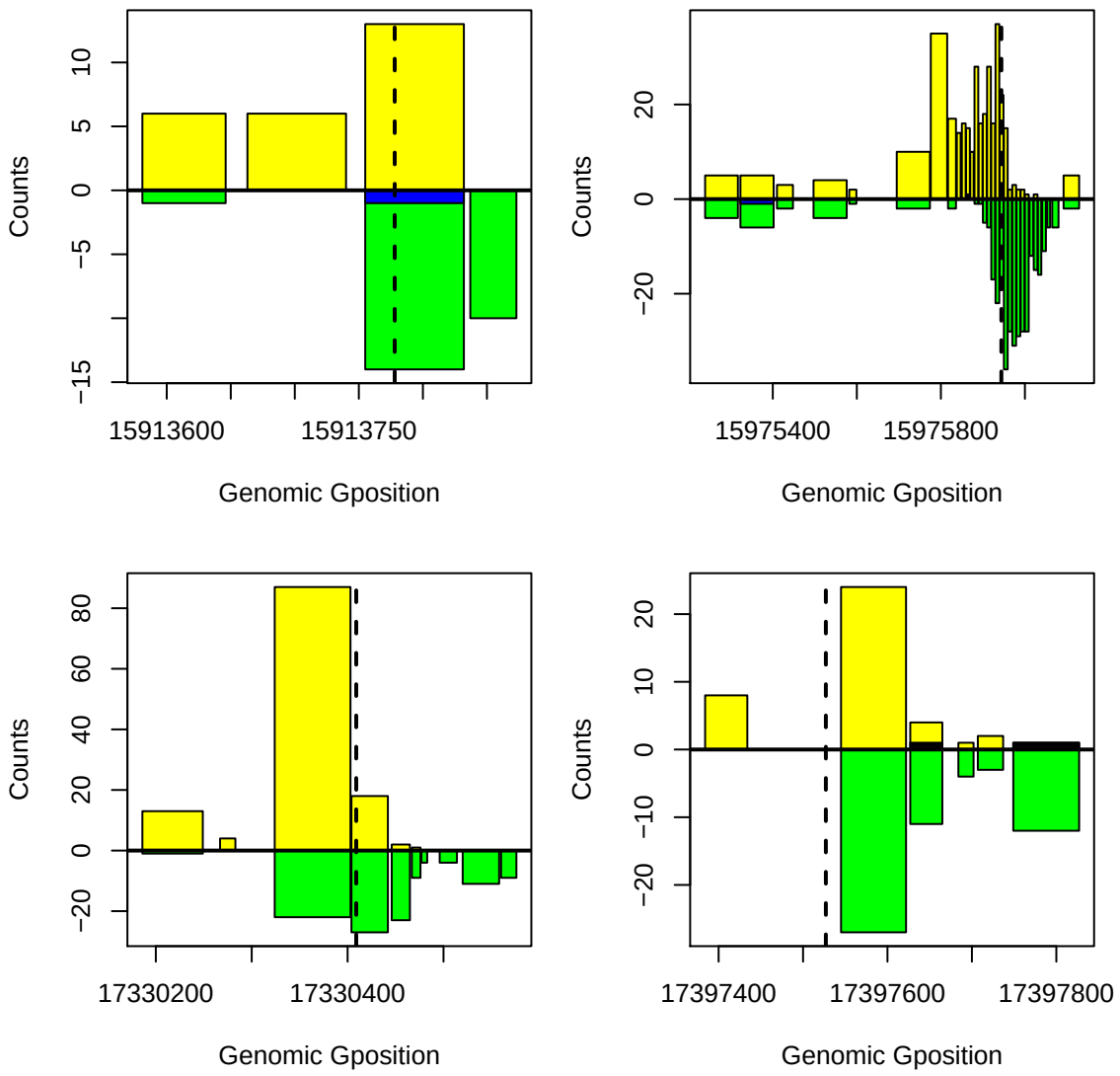
```
> reg1 = peakreg(chrpos = tagct22[, 1:3], count = (tagct22[, 5:6] -
+     tagct22[, 7:8]), pp = res1$pp, cutoff = 0.5, method = "ppcut",
```

```
+     maxgap = 200)
> print(dim(reg1))

[1] 84 11

> print(reg1[1:3, ])

    chr   gstart     gend rstart rend  peakpos meanpp ct1 ct2 ct12  sym
1 chr22 15913581 15913873    634  637 15913778   0.96  25  24   49 0.49
2 chr22 15975239 15976129    797  828 15975944   0.98 331 339  670 0.49
3 chr22 17330186 17330576   2568 2577 17330409   0.83 125 110  235 0.47
```

Note, the 5' positions of the sequence tags are used as the genomic positions for the NRSF data. To infer the actual binding sites, one may add 13 bp to the peak position (reg1$peakpos + 13) because the tags' length is 25 bp. If the middle positions of the sequence tags are used as the genomic positions, the user doesn't need to do the adjustment.

Plot some enriched regions. The dash lines indicate the region center, usually the true binding sites.

```
> par(mfrow = c(2, 2), mar = c(4.1, 4.1, 2, 1))
> for (i in 1:4) {
+     ID = (reg1[i, 4]):(reg1[i, 5])
+     plotreg(tagct22[ID, 2:3], tagct22[ID, 5:6], tagct22[ID, 7:8],
+         peak = reg1[i, 6])
+ }
```



Call the enriched regions using 0.05 FDR cutoff. The FDR is calculated using a direct posterior probability approach (Newton et al., 2004).

```
> reg2 = peakreg(tagct22[, 1:3], tagct22[, 5:6] - tagct22[, 7:8],
+     res1$pp, 0.05, method = "fdrcut", maxgap = 200)
> print(dim(reg2))
```

```
[1] 93 11
```

```
> print(reg2[1:3, ])
```

```
    chr   gstart     gend rstart rend  peakpos meanpp ct1 ct2 ct12  sym
1 chr22 15913581 15913873    634  637 15913778   0.96  25  24   49 0.49
2 chr22 15975239 15976129    797  828 15975944   0.98 331 339  670 0.49
3 chr22 17330186 17330617   2568 2578 17330409   0.79 125 116  241 0.48
```

The columns 1-3 of the enriched regions (e.g. reg2[,1:3]) can be used to extract the sequences from the UCSC genome browser. Alternatively, one may create a BED file using the peak position of the enriched regions. For example,

```
> bed = data.frame(chr = reg2[, 1], gstart = reg2[, 6] - 100, gend = reg2[,
+     6] + 100)
```

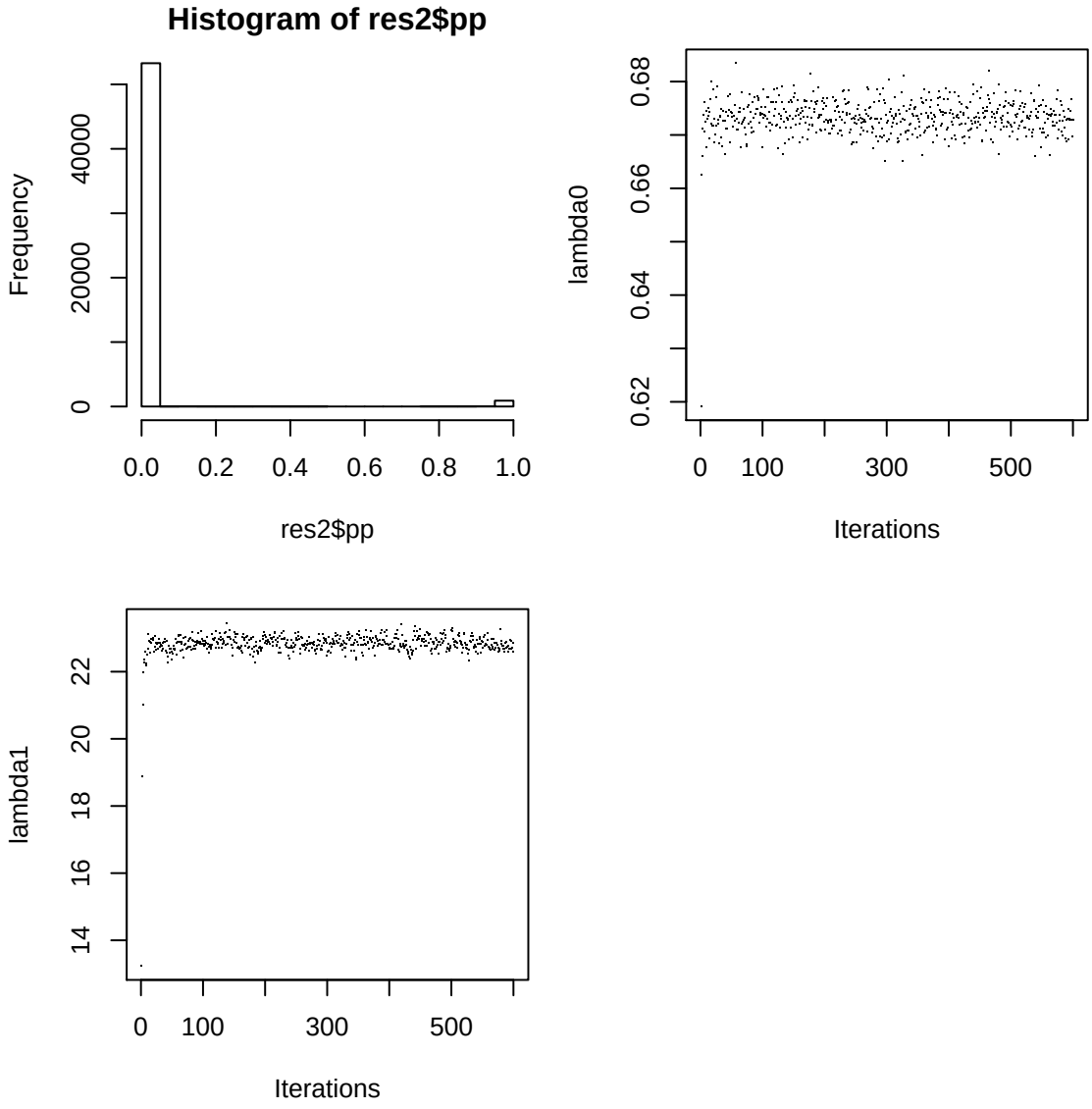
3.4 Model dynamic signal profiles using function 'iSeq2'

The latent variable X can be modeled through a high-order Ising model using function iSeq2. The interaction parameter k for the high-order (or the standard/first-order) Ising model is fixed and set by the user in iSeq2. To apply the second-order Ising model to ChIP-seq data, the user can let winsize = 2. If set winsize = 1, it will be the standard Ising model. To use a high-order Ising model, according to our experience, a balance between high sensitivity and low FDR can be achieved when winsize = 2. The critical value for the second-order Ising model is about 1.0. In general, increasing the value of k will lead to less enriched regions, which amounts to setting a stringent criterion for detecting enriched regions.
Model the NRSF data using the second-order Ising model.

```
> res2 = iSeq2(Y = tagct22[, 1:4], gap = 200, burnin = 100, sampling = 500,
+     winsize = 2, ctcut = 0.95, a0 = 1, b0 = 1, a1 = 5, b1 = 1,
+     k = 1, verbose = FALSE)
```

Plot the model parameters to see whether they converge. If the chains do not mix well, one can adjust the parameter **k** and/or normsd to see how they affect the results.

```
> par(mfrow = c(2, 2), mar = c(4.1, 4.1, 2, 1))
> hist(res2$pp)
> plot(res2$lambda0, pch = ".", xlab = "Iterations", ylab = "lambda0")
> plot(res2$lambda1, pch = ".", xlab = "Iterations", ylab = "lambda1")
```

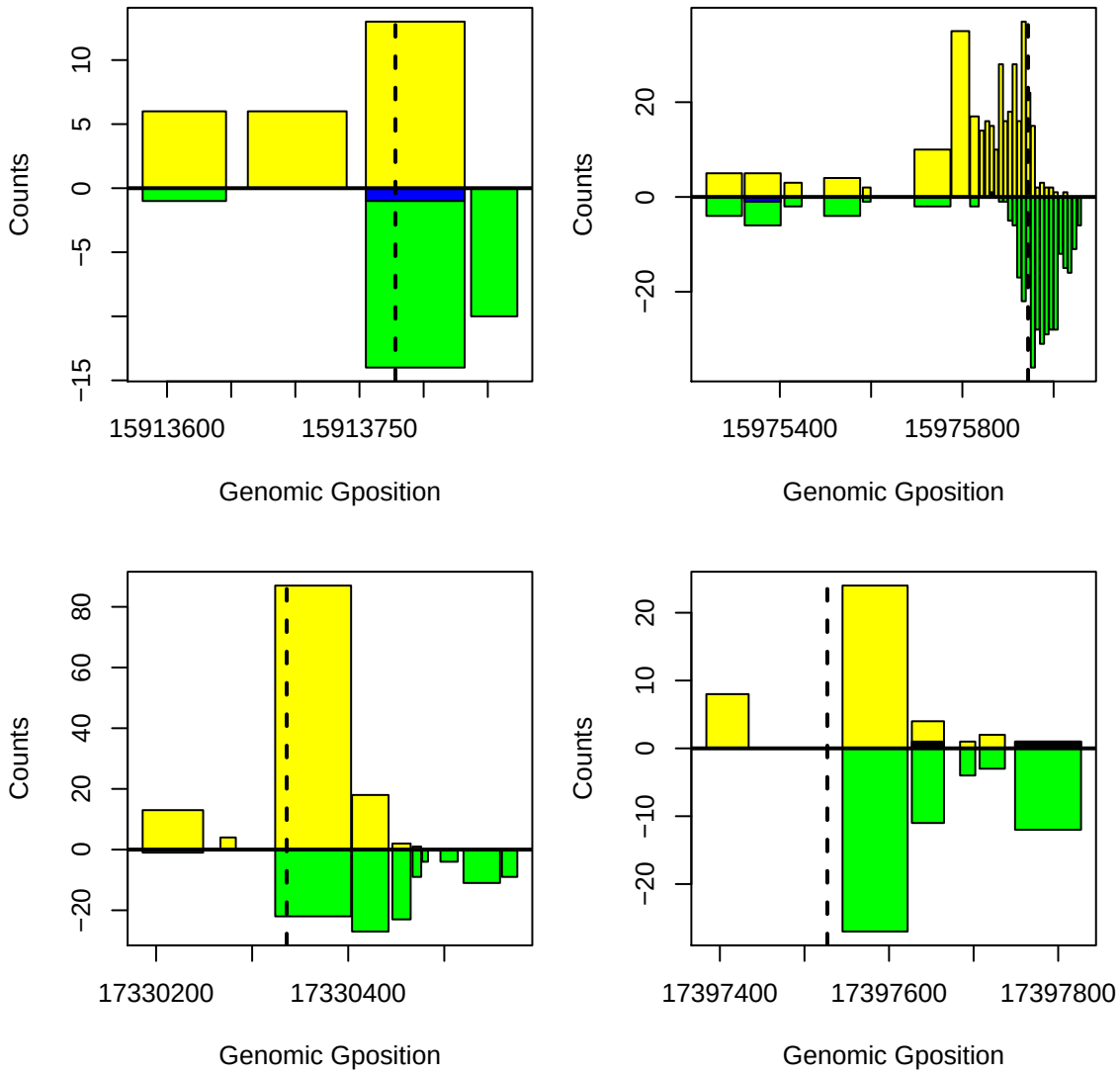


3.5 Call enriched regions detected by iSeq2 using function 'peakreg'

```
> reg2 = peakreg(tagct22[, 1:3], tagct22[, 5:6], res2$pp, 0.5,
+     method = "ppcut", maxgap = 200)
> print(dim(reg2))

[1] 93 11

> print(reg2[1:3, ])

    chr   gstart     gend rstart rend  peakpos meanpp ct1 ct2 ct12  sym
1 chr22 15913581 15913873    634  637 15913778   0.88  25  25   50 0.50
2 chr22 15975239 15976060    797  826 15975944   0.93 327 332  659 0.50
3 chr22 17330186 17330576   2568 2577 17330336   0.70 125 110  235 0.47
```

Plot some enriched regions detected by iSeq2.

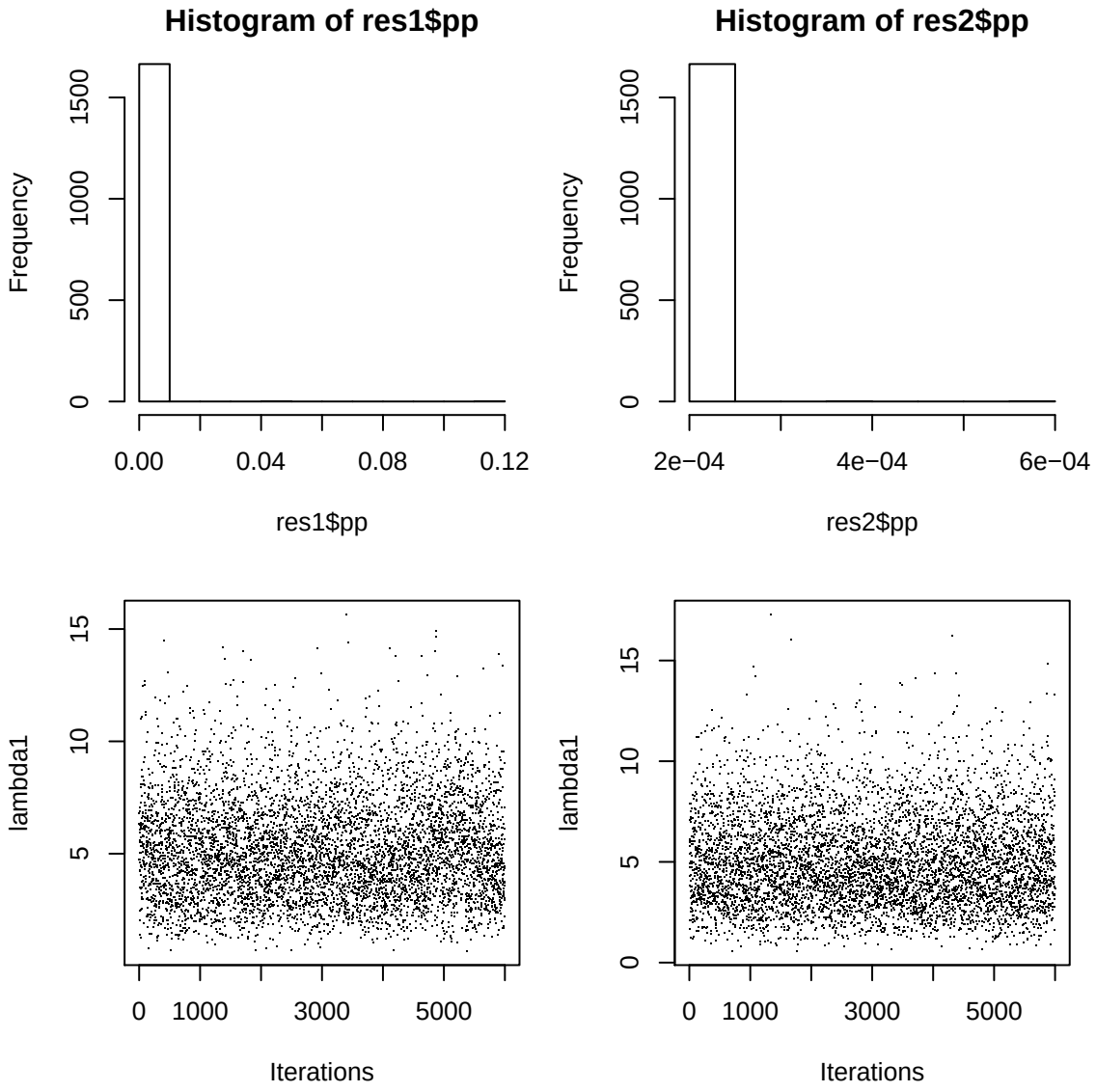
```
> par(mfrow = c(2, 2), mar = c(4.1, 4.1, 2, 1))
> for (i in 1:4) {
+     ID = (reg2[i, 4]):(reg2[i, 5])
+     plotreg(tagct22[ID, 2:3], tagct22[ID, 5:6], tagct22[ID, 7:8],
+         peak = reg2[i, 6])
+ }
```



4 Tips

Finally, let's analyze the chromosome Y data. Intuitively, it seems that there is no binding region on chromosome Y.

```
> tagctY = tagct[tagct[, 1] == "chrY", ]
> print(table(tagctY[, 4]))

   0    1    2    3    4    5    6    7    9
 435 1006  140   52   21    4    7    1    1

> res1 = iSeq1(Y = tagctY[, 1:4], gap = 200, burnin = 1000, sampling = 5000,
+     ctcut = 0.95, a0 = 1, b0 = 1, a1 = 5, b1 = 1, k0 = 3, mink = 0,
+     maxk = 10, normsd = 0.5, verbose = FALSE)

Warning: all bins are in the same state at the last MCMC iteration.
 NO enriched region is found!

> res2 = iSeq2(Y = tagctY[, 1:4], gap = 200, burnin = 1000, sampling = 5000,
+     winsize = 2, ctcut = 0.95, a0 = 1, b0 = 1, a1 = 5, b1 = 1,
+     k = 3, verbose = FALSE)

Warning: all bins are in the same state at the last MCMC iteration.
 NO enriched region is found!

> par(mfcol = c(2, 2), mar = c(4.1, 4.1, 2, 1))
> hist(res1$pp)
> plot(res1$lambda1, pch = ".", xlab = "Iterations", ylab = "lambda1")
> hist(res2$pp)
> plot(res2$lambda1, pch = ".", xlab = "Iterations", ylab = "lambda1")
```



In this case, all the bins are only in the non-enriched state and the posterior probabilities are zero or close to zero. In some cases, if the λ_1 is small (e.g. $\approx a_1/b_1$, the mean value of the gamma prior), it suggests that there is no enriched region on that chromosome. The user may not claim any enriched regions found on that chromosome, even if some posterior probabilities are high.

According to my experience, iSeq1 methods work well for most of ChIP-seq data I have analyzed. For very noisy data, iSeq1 may not work. The parameter λ_1 estimates for very noisy data are usually less than 1 and the posterior probabilities are not dichotomized and dominated by 0, which suggest that the Ising model is not in the super-paramagnetic phase. Only the super-paramagnetic phase reflects the binding events on the chromosomes. Therefore, if iSeq1 doesn't work, the user can use iSeq2. In iSeq2 function, the interaction parameter k is fixed by the user. The user can increase the value of k to let the phase transition occur so that the Ising model reaches the super-paramagnetic phase.

5 Parallel Computation

To speed up the analysis, the user can do parallel computation easily. The user needs to install the snow and snowfall packages. The following is an example.

library(snow)
library(snowfall)
dataList = list(chr22=tagct22,chrY=tagctY)
sfInit(parallel=TRUE,cpus=2,type="SOCK")
res=sfLapply(dataList,iSeq1,gap=200,burnin=100,sampling=200,ctcut=0.95,a0=1,b0=1,a1=1,b1=1,
k0=3,mink=0,maxk=10,normsd=0.1,verbose=FALSE)

6 Citing iSeq

If you use iSeq package, please cite "Mo, Q. (2010). A fully Bayesian hidden Ising model for ChIP-seq data analysis (Submitted).".

```
> sessionInfo()

R version 2.13.0 (2011-04-13)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=C              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

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] iSeq_1.4.0

loaded via a namespace (and not attached):
[1] tools_2.13.0
```

References

Mo, Q., Liang, F. (2010a). Bayesian Modeling of ChIP-chip data through a high-order Ising Model. *Biometrics* **66**, 1284-1294.

Mo, Q., Liang, F. (2010b). A hidden Ising model for ChIP-chip data analysis. *Bioinformatics* **26(6)**, 777-783.

Mo, Q. (2010). A fully Bayesian hidden Ising model for ChIP-seq data analysis. (Submitted).

Newton, M., Noueiry, A., Sarkar, D., Ahlquist, P. (2004). Detecting differential gene expression with a semiparametric hierarchical mixture method. *Biostatistics* **5** , 155-176.

Johnson, D.S., Mortazavi, A., Myers, R.M., Wold, B. (2007). Genome-wide mapping of in vivo protein-DNA interactions. *Science* **316**, 1497-1502.