The MeDiChI ChIP-chip deconvolution package

David J Reiss, Institute for Systems Biology

October 6, 2010

Chromatin immunoprecipitation followed by hybridization to a genomic tiling microarray (ChIP-Chip) is a routinely used protocol for localizing the genomic targets of DNA-binding proteins. The resolution to which binding sites in this assay can be identified is commonly considered to be limited by two factors: (a) the resolution at which the genomic targets are tiled in the microarray, and (b) the large and variable lengths of the immunoprecipitated DNA fragments.

The MeDiChI package uses a generative model of binding sites in ChIP-chip data, and an approach for efficiently and robustly learning that model from diverse data sets. We have evaluated MeDiChI's performance using simulated data, as well as on several diverse ChIP-chip data sets collected on widely different tiling array platforms for two different organisms (*S. cerevisiae* and *H. salinarum NRC-1*). We find that MeDiChI accurately predicts binding locations to a resolution greater than that of the probe spacing, even for overlapping peaks, and can increase the effective resolution of tiling array data by a factor of 5× or better. Moreover, the method's performance on simulated data provides insights into effectively optimizing the experimental design for increased binding site localization accuracy and efficacy.

The MeDiChI package includes functions for constructing a peak profile, using it to deconvolve the ChIP-chip data, and plotting and examining the resulting fits. It also includes functions for generating simulated data for quality assessment.

We encourage you to read this package's publication, listed at the bottom of this document, and to visit the companion website, itemized just below.

`http://baliga.systemsbiology.net/medichi`

1 Technical Background and Notes

The MeDiChI package requires a very basic data format, by default a two-column matrix listing the central chromosomal coordinate of each probe and its intensity (*NOTE: intensities should be un-logged*). Multiple replicates may be included in this single data set. For more information, see the online help documentation for this package.

2 Demo 1: Load and deconvolve a small chromosomal segment of ChIP-chip data

In this first demonstration, we

- Initialize the MeDiChI library and load the example low-resolution *H. salinarum NRC-1* data set
- Deconvolve a small part of this data set on the main chromosome, including 100 bootstraps
- Plot and examine the resulting fit and bootstrap distribution

```
> library(MeDiChI)

Loading MeDiChI, version 0.3.9 (Wed Oct  6 13:46:32 2010)

> data("halo.lowres", package = "MeDiChI")
> fit <- chip.deconv(data.halo.lowres, where = "Chr", fit.res = 10,
+     center = 650000, wind = 20000, max.steps = 100, n.boot = 10,
+     kernel = kernel.halo.lowres, verbose = TRUE)

Using 1.520822 as data cutoff!
MEAN PROBE SPACING = 500
Step for min AIC: 33 15 ; BIC: 26 14 ; using: bic
13 coeffs at > 0 for LARS step 26
Number of coeffs: 13 ... Reduced to 8 non-redundant coeffs.
*** BOOTSTRAP ITER: 2 ***
Step for min AIC: 10 8 ; BIC: 1 1 ; using: bic
0 coeffs at > 0 for LARS step 1
Number of coeffs: 0 ... Reduced to 0 non-redundant coeffs.
*** BOOTSTRAP ITER: 3 ***
Step for min AIC: 5 5 ; BIC: 3 3 ; using: bic
2 coeffs at > 0 for LARS step 3
Number of coeffs: 2 ... Reduced to 2 non-redundant coeffs.
*** BOOTSTRAP ITER: 4 ***
Step for min AIC: 5 5 ; BIC: 4 4 ; using: bic
3 coeffs at > 0 for LARS step 4
Number of coeffs: 3 ... Reduced to 2 non-redundant coeffs.
*** BOOTSTRAP ITER: 5 ***
Step for min AIC: 4 4 ; BIC: 2 2 ; using: bic
1 coeffs at > 0 for LARS step 2
Number of coeffs: 1 ... Reduced to 1 non-redundant coeffs.
```

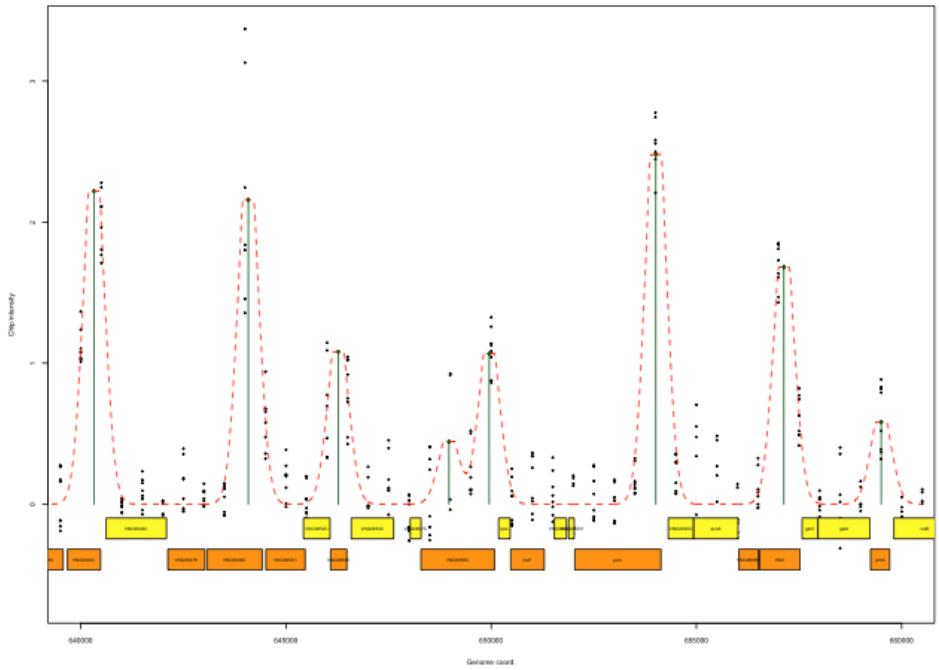
```
*** BOOTSTRAP ITER: 6 ***
Step for min AIC: 4 4 ; BIC: 2 2 ; using: bic
1 coeffs at > 0 for LARS step 2
Number of coeffs: 1 ... Reduced to 1 non-redundant coeffs.
*** BOOTSTRAP ITER: 7 ***
Step for min AIC: 5 5 ; BIC: 4 4 ; using: bic
3 coeffs at > 0 for LARS step 4
Number of coeffs: 3 ... Reduced to 2 non-redundant coeffs.
*** BOOTSTRAP ITER: 8 ***
Step for min AIC: 6 6 ; BIC: 2 2 ; using: bic
1 coeffs at > 0 for LARS step 2
Number of coeffs: 1 ... Reduced to 1 non-redundant coeffs.
*** BOOTSTRAP ITER: 9 ***
Step for min AIC: 7 7 ; BIC: 1 1 ; using: bic
0 coeffs at > 0 for LARS step 1
Number of coeffs: 0 ... Reduced to 0 non-redundant coeffs.
*** BOOTSTRAP ITER: 10 ***
Step for min AIC: 6 6 ; BIC: 2 2 ; using: bic
1 coeffs at > 0 for LARS step 2
Number of coeffs: 1 ... Reduced to 1 non-redundant coeffs.
```

```
> coef(fit)
```

```
    position intensity p.value <
83    640322 2.2199189       0.1
210   644082 2.1581575       0.1
380   646272 1.0803201       0.1
501   648972 0.4423681       0.7
599   649952 1.0666237       0.1
706   654012 2.4796942       0.1
819   657132 1.6817031       0.1
907   659502 0.5821610       0.4
```

```
> plot(fit, plot.genes = TRUE, cex = 0.5, cex.lab = 0.8, cex.axis = 0.8)
```

You should see a plot of the chunk of data and the resulting fit appear, similar to this one:



3 Demo 2: Deconvolve an entire data set

Here we deconvolve the entire data set rather than one small chunk. This method effectively deconvolves small chunks of data in a scrolling window and merges the results at the end. Depending on the size of the data set, this procedure can take a while (this particular example will take about 15 minutes). In this demonstration, we

- Deconvolve the entire data set, including bootstraps
- Plot and examine parts of the resulting fit and bootstrap distribution

```
> fits <- deconv.entire.genome(data.halo.lowres, max.steps = 100,
+     fit.res = 30, n.boot = 10, boot.sample = "residual", kernel = kernel.halo.lowre
+     verbose = FALSE)
> coef(fits$fits.fin$Chr)
> plot(fits$fits.fin$Chr, center = 650000, wind = 20000, plot.genes = T)
```

You should see a plot appear, similar to the one above. The print-out of the coefficients will include p-values for each detected peak (based upon bootstrap statistics across the entire data set).

4 Demo 3: Learn the peak profile from data

In the previous twoexamples, we used a pre-computed ChIP-chip peak profile (the same one that was used in the manuscript [1]. In this demo, we will show how this peak profile is computed from the data set. This method starts with a guess for good starting parameters, generates a binding profile using those parameters, and detects binding sites across the data set as in the previous example. It then identifies the 20 brightest, most isolated peaks, and adjusts the peak profile parameters to better reflect those peaks. This process is repeated several times. It can be sped up by adjusting the parameters of the following function to make its repeated calls to `deconv.entire.genome()` execute faster. In this demonstration, we

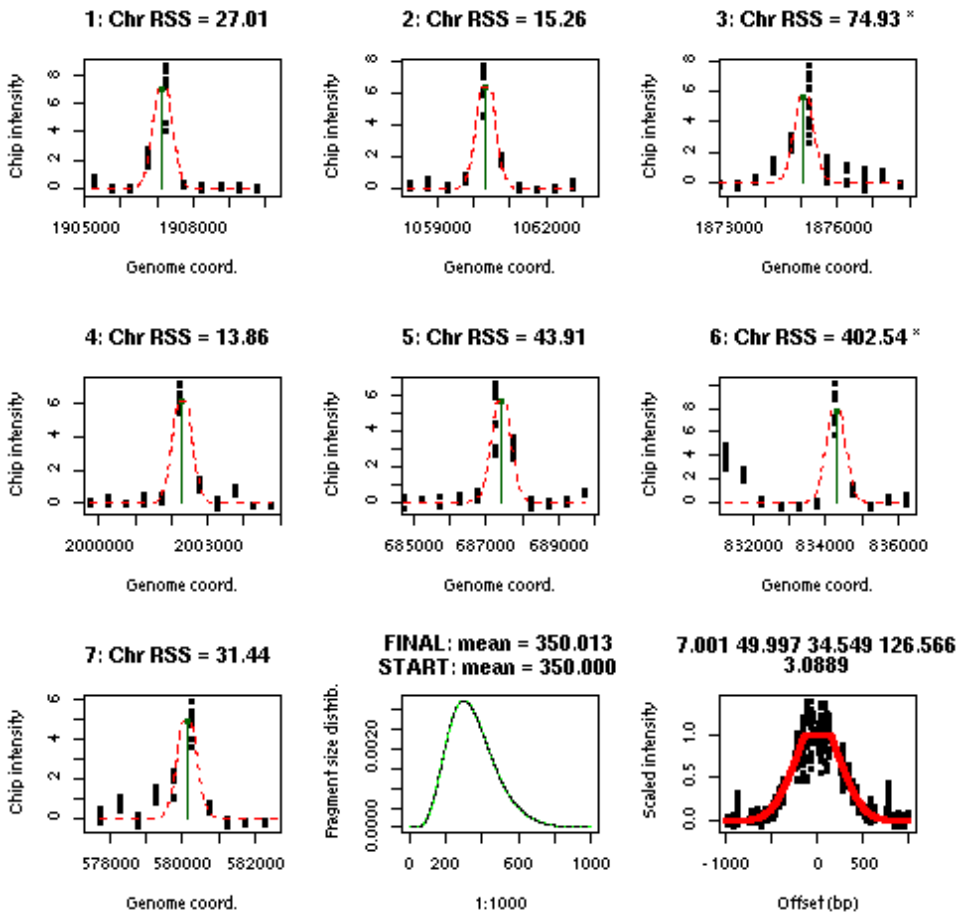
- Fit a peak profile (kernel) to the low-resolution *H. salinarum NRC-1* data set
- Plot the resulting fit and learn how to use the output for a run of `chip.deconv()` or `deconv.entire.genome()`

```
> peak.params <- fit.peak.profile(data.halo.lowres, tile.size = 500,
+     quant.cutoff = "q0.98", chrom = "Chr", fit.res = 30, max.steps = 100,
+     plot = T, name = "Halo-lowres")
```

The resulting structure can be plotted, which will display seven of the brightest peaks contributing to the fit, the best-fit DNA fragment length distribution, and the final peak profile, including all data data points surrounding the 20 brightest, isolated peaks (shifted and normalized based on their best-fit positions and intensities).

```
> plot(peak.params)
```

This results in something like this for the above example:



`peak.params`*kernel may be used as a kernel for more accurate deconvolution* `chip.deconv()` *or* `deconv.ent`

5 For More Information

For more information, please visit:
http://baliga.systemsbiology.net/medichi

6 References

- Reiss, DJ and Facciotti, MT and Baliga, NS. (2007). "Model-based deconvolution of genome-wide DNA binding", *Bioinformatics*. doi: 10.1093/bioinformatics/btm592.