A Bayesian Linear Model of Multiple High-throughput Sequencing Data under Unknown Environmental Conditions

Yuan Yin

Abstract

The regulatory relationship between transcription factors (TF) and target gene (TG) can alter under different environmental conditions, and it is typically hidden to us what experiment conditions correspond to which regulatory relationship. Therefore, even with a large number of public RNASeq data labeled with experiment conditions, uncovering the regulatory relationships are not straightforward. ChIPSeq experiments inform us the TFs that bind near TG. For ChIPSeq setup that artificially induces the production of a TF and saturate cells with the TF, we may assume those TFs that bind near TG represents the entire regulator space.

If we treat each TF and TG as a random variable, we may summarize the data we have as follows: an unlabeled collection of instantiations of random variables from multiple unknown data generating processes, although we do know all the TFs whose (unknown) subset participated in each data generating process. We present a statistical model to tackle this problem: find clusters in RNASeq data, select true regulators from TFs in ChIPSeq data for each cluster using a spike-and-slab prior strategy, and run a linear regression of TG vs. true regulators in each cluster. These subtasks are built in a single model, and we develop a Metropolis-within-Gibbs sampling strategy for inference on the parameters of the model.

Section 1: Introduction

The accessible public databases such as Gene Expression Omnibus provides massive amount of RNASeq data that enable us to ask some fundamental questions, such as how genes are regulated in organisms. RNASeq is typically thought of as a “snapshot” of RNA in cells at the time the sample is taken. The compendium of RNASeq data in public databases, generated under diverse growth conditions and experimental treatments, can be thought of as a “photo album” of the gene dynamics of organisms. This provides plenty avenues in modeling and understanding the complex regulatory relationships among genes. This project aims at tackling this task in *E. coli*, where we try to establish a mapping between the activity level of regulators and target gene.

RNASeq data are typically used as a proxy to the gene activity level. At the molecular level, when a gene is about to be activated, transcription machinery is recruited to the gene, RNA copies are then being made, and the abundance of the RNA copies are captured by the RNASeq. It is from the RNASeq data that we infer the gene is being activated.

However, there are caveats in using RNASeq data to infer the activity level of transcription regulators. The regulators, which are part of the transcription machinery, participated in the gene regulation as proteins, not RNA. And hence their form and activity level when they regulate the genes are simply not reflected by the RNASeq data. As an example, SoxR is a regulator that responds to oxidative stress. Its active form requires Fe-S clusters, without which the regulator shows no regulatory activity even though it binds to the promoter of its regulated genes. Furthermore, those SoxR with clusters can have its activity level altered by the redox state of the clusters. It is clear in this example that RNA level is not a good proxy to SoxR activity level.

Because the protein level of a regulator is not directly observed, we may use their RNA level as a proxy to their activity level under the hypothesis that at different conditions, there is a different mapping between the RNA level and the protein activity level. More specifically, we assume that regulators have two states, an ON and an OFF state, and each RNASeq sample is collected when each regulator takes either the ON or the OFF state. Under the hypothesis, we build a regression model of target genes and potential regulators, and use Bayesian inference to validate the hypothesis.

The report is divided into the following sections. In section 2, we describe our model and the reasoning for its design. In section 3, we describe the inference algorithm. In section 4, we run our algorithm on a simulated dataset and discuss on our findings.

Section 2: Model Description

Our first attempt to model the gene regulation is to build a linear regression model of transcription factors (TFs) and the target gene. To identify potential regulators (regressors), we have conducted ChIPSeq experiment on the TFs. ChIPSeq data reveals if and to what extent does each TF bind to the target gene, and we shall assume that all the TFs that bind to the gene are potentially its regulators. This information reduces the number of potential regulators from all TFs (about 300) to less than a few dozens. We index regressors with j=1, 2, …, J. We index samples in our dataset with i=1, 2, …, N.

As we have stated earlier, we assume that each TF takes either an ON or an OFF state under each condition. We index the conditions using k. So, for J number of TFs, k=1, 2, …, 2^J. The exponential growth of k is likely to create problems for the inference, as our dataset could quickly be overfitted by the exponential number of parameters. To proceed with modeling, we tweak the problem by assuming that there is only a finite K conditions, i.e. k=1, 2, …, K. With this, we let e_i be a random variable that indicates if sample i is generated from condition k:

$$e_i \sim Categorical(\lambda_1, \ldots, \lambda_K)$$

$\boldsymbol{e} \in \{1,2,\ldots,K\}^N$ is therefore a vector of length N, indicating which condition each of the N samples are generated from.

For $\lambda's$, because we don't have any prior knowledge on the distribution of conditions, we model them using a low variance Dirichlet distribution:

$$\boldsymbol{\lambda} \sim Dirichlet(1.5)$$

Where $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_K]$.

Now we consider the regressors in the regression. Even though we have reduced the feature space thanks to the ChIPSeq, not all TFs selected by ChIPSeq are true regulators. Biologically, some TF may have an affinity to a binding region that enables its binding near the gene, even though it is not regulating the gene. To account for this, we define Bernoulli random variables r_{kj} to indicate that TF_j is a true regulator under condition k:

$$r_{kj} \sim Bernoulli(\gamma_{kj})$$

And we adopt a spike-and-slab prior for our regression coefficients as our feature selection strategy:

$$\boldsymbol{\beta_k}|\boldsymbol{r_k}, c_k \sim N\left(\mathbf{0}, \left[1, c_k^{1-r_{k1}}, \dots, c_k^{1-r_{kJ}}\right] \circ s^2\boldsymbol{I_{J+1}}\right) = N\left(\begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}, \begin{bmatrix} s^2 & 0 & \dots & 0 \\ 0 & c_k^{1-r_{k1}}s^2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & c_k^{1-r_{kJ}}s^2 \end{bmatrix}\right)$$

Where c_k is a positive random variable that is much smaller than 1, and s^2 is a big positive number. The product of c_k and s^2 is designed to be small. Intuitively, when r_{kj} takes 0, i.e. the jth regressor is NOT a true regulator, then $\beta_{kj} \sim N(0, c_k s^2)$. Since the variance, $c_k s^2$, is small, the instantiation of β_{kj} will be very close to 0.

Some readers may be disturbed by the double negative design choice of the spike-and-slab prior. We did this to ensure that c_k can be modeled using an Inverse-Gamma distribution, which is conjugate to the distribution of $\boldsymbol{\beta_k}$, and its posterior will take a closed form.

For the parameter of r_{kj}, denoted as γ_{kj}, we model it using beta distribution:

$$\gamma_{kj} \sim Beta(a_j, b_j)$$

For its hyperparameters, it is an empirical belief that in *E. coli*, the average number of regulators of a gene is around 2. To reflect this belief, we use the restraint:

$$\sum_j \frac{a_j}{a_j + b_j} = 2$$

This is because the sum of r_{kj} over j, which is the total number of TFs being true regulator, would have an expectation of $\sum_j \gamma_{kj}$, and the expectation of γ_{kj} is $\frac{a_j}{a_j+b_j}$.

In addition to empirical belief, in ChIPSeq experiment, TF binding enrichment (Δ_j), which could reflect the binding affinity between a TF and a target gene, could encode information on how likely a TF is a true regulator. So we would like γ_{kj} to be higher for TF that has higher enrichment. If we define the expectation of γ_{kj} to be proportional to the enrichment:

$$E(\gamma_{kj}) = \frac{a_j}{a_j + b_j} = c \cdot \Delta_j$$

We can solve for c in terms of Δ_j, and get:

$$\frac{a_j}{a_j + b_j} = \frac{2\Delta_j}{\sum_j \Delta_j}$$

We would like to keep this prior as relatively uninformative, so we set $a_j = 1.5$, and we can solve for b_j:

$$b_j = \frac{1.5(\sum_j \Delta_j - 2\Delta_j)}{2\Delta_j}$$

c_k is a small scaling factor. We use an Inverse-Gamma distribution with parameter of $g_1 = 100$ and $g_2 = 1$ to model it:

$$c_k \sim IG(g_1, g_2)$$

Finally, for the linear regression model, we use the classic one with Gaussian noise:

$$y_i | e_i = k \ \sim \ N\left(\boldsymbol{X_i}\boldsymbol{\beta_k}, \sigma_k^2\right)$$

Where $\boldsymbol{X_i}$ is the i^th sample of the data, and $\boldsymbol{\beta_k}$ is the vector of regression coefficient under environment k.

We set the prior of the noise term using an Inverse-Gamma distribution with parameters $m_1 = 1$ and $m_2 = 1$:

$$\sigma_k^2 \sim IG(m_1, m_2)$$

To summarize, let $\boldsymbol{\theta_k} = [\boldsymbol{\beta_k}, \sigma_k^2, \boldsymbol{r_k}, \boldsymbol{\gamma_k}, c_k]$ be all parameters that are dependent upon condition k, we have a hierarchical model:

$$P(\boldsymbol{\theta}, \boldsymbol{e}, \boldsymbol{\lambda} | \boldsymbol{y}) = P(\boldsymbol{\lambda})P(\boldsymbol{e}|\boldsymbol{\lambda})P(\boldsymbol{\theta}|\boldsymbol{e})P(\boldsymbol{y}|\boldsymbol{\theta}) = P(\boldsymbol{\lambda})P(\boldsymbol{e}|\boldsymbol{\lambda})\prod_k P(\boldsymbol{y_{i:e_i=k}}|\boldsymbol{\theta_k})P(\boldsymbol{\theta_k})$$

Graphically, the parameter dependencies are as follows, where the parameters in dashes box are dependent on condition variable e.

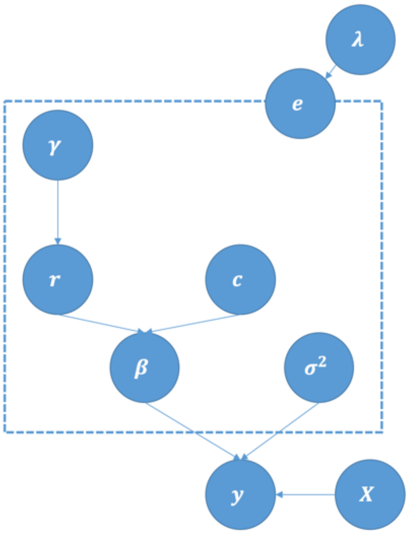

Figure 1. Summary of model parameters.

Section 3: Inference Algorithm

The model is designed in such a way that the full conditional distributions are mostly in closed form, with which it is convenient to use a Gibbs sampling-based method for inference. We have derived the full conditional distributions in the appendix. Because r_{kj} does not have a closed form conditional distribution, we use a Metropolis-within-Gibbs algorithm for inference. Here the superscript (l) indicates the l^{th} sample from the sampler.

Algorithm:

Given $\boldsymbol{\lambda}^{(l)} \in \{\mathbb{R}_+^K : \sum_k \lambda_k = 1\}$, $\boldsymbol{e}^{(l)} \in \{1,2,\ldots,K\}^N$, $\boldsymbol{\beta}^{(l)} \in \mathbb{R}^{K\times(J+1)}$**,** $\sigma^{2(l)} \in \mathbb{R}_+^K$, $\boldsymbol{\gamma}^{(l)} \in (0,1)^{K\times J}$, $\boldsymbol{r}^{(l)} \in \{0,1\}^{K\times J}$, $c^{(l)} \in \mathbb{R}_+^K$, iteratively update each parameter:

- Count number of samples from condition k=1, 2, …, K, denoted as $\left|i: e_i^{(l)} = k\right|$
- Sample $\boldsymbol{\lambda}^{(l+1)}$ from
$$Dirichlet\left(1.5 + \left|i: e_i^{(l)} = 1\right|, \ldots, 1.5 + \left|i: e_i^{(l)} = K\right|\right)$$
- For i=1, 2, …, N, sample $e_i^{(l+1)}$ from
$$Categorical\left(\lambda_1^{(l+1)}\left(\sigma_1^{2(l)}\right)^{-\frac{1}{2}}\exp\left\{-\frac{\left(y_i - \boldsymbol{X}_{i,:}\boldsymbol{\beta}_{1,:}^{(l)}\right)^2}{2\sigma_1^{2(l)}}\right\}, \ldots, \lambda_K^{(l+1)}\left(\sigma_K^{2(l)}\right)^{-\frac{1}{2}}\exp\left\{-\frac{\left(y_i - \boldsymbol{X}_{i,:}\boldsymbol{\beta}_{K,:}^{(l)}\right)^2}{2\sigma_K^{2(l)}}\right\}\right)$$
- For k=1, 2, …, K, and j=1, 2, …, J, sample $\gamma_{kj}^{(l+1)}$ from
$$Beta\left(a_j + r_{kj}^{(l)}, b_j - r_{kj}^{(l)} + 1\right)$$
- For k=1, 2, …, K, and j=1, 2, …, J, propose sample $\bar{r}_{kj}$from Bernoulli(0.5), and u from Uniform(0,1). If $u < \min\left(1, \frac{\tilde{\pi}(\bar{r}_{kj})}{\tilde{\pi}\left(r_{kj}^{(l)}\right)}\right)$, $r_{kj}^{(l+1)} = \bar{r}_{kj}$; otherwise $r_{kj}^{(l+1)} = r_{kj}^{(l)}$.

  Here $\tilde{\pi}(r_{kj}) = \left(\gamma_{kj}^{(l+1)}\right)^{r_{kj}+a_j-1}\left(1-\gamma_{kj}^{(l+1)}\right)^{b_j - r_{kj}}\left(c_k^{(l)}\right)^{-\frac{1-r_{kj}}{2}}\exp\left\{-\frac{\left(\beta_{kj}^{(l)}\right)^2}{2\left(c_k^{(l)}\right)^{1-r_{kj}}s^2}\right\}$
- For k=1, 2, …, K, sample $c_k^{(l+1)}$ from
$$IG\left(g_1 + \frac{J - \sum_j r_{kj}^{(l+1)}}{2}, g_2 + \frac{\sum_{j:r_{kj}^{(l+1)}=0}\left(\beta_{kj}^{(l)}\right)^2}{2s^2}\right)$$
- For k=1, 2, …, K, sample $\boldsymbol{\beta}_{k,:}^{(l+1)}$ from
$$N\left(\frac{1}{\sigma_k^{2(l)}}\left(\boldsymbol{\Sigma}^{-\mathbf{1}} + \frac{\boldsymbol{X^T X}}{\sigma_k^{2(l)}}\right)^{-\mathbf{1}}\boldsymbol{X^T y}, \left(\boldsymbol{\Sigma}^{-\mathbf{1}} + \frac{\boldsymbol{X^T X}}{\sigma_k^{2(l)}}\right)^{-\mathbf{1}}\right)$$

  where $\boldsymbol{\Sigma} = \left[1, \left(c_k^{(l+1)}\right)^{1-r_{k1}}, \ldots, \left(c_k^{(l+1)}\right)^{1-r_{kJ}}\right] \circ s^2\boldsymbol{I_{J+1}}$, $\boldsymbol{y} = \boldsymbol{y}_{i:e_i^{(l+1)}=k}$, $\boldsymbol{X} = \boldsymbol{X}_{i:e_i^{(l+1)}=k}$
- For k=1, 2, …, K, sample $\sigma_k^{2(l+1)}$ from
$$IG\left(m_1 + \frac{\left|i: e_i^{(l+1)} = k\right|}{2}, m_2 + \frac{\left|\boldsymbol{y}_{\boldsymbol{i}:e_i^{(l+1)}=k} - \boldsymbol{X}_{\boldsymbol{i}:e_i^{(l+1)}=k}\boldsymbol{\beta}_{k,:}^{(l+1)}\right|_2^2}{2}\right)$$

Section 4: Discussions

We test our model first with a single component to ensure our algorithm and programs work as expected. We simulated data with a single mixing component, and fit it with our model. As shown in the following figures. The trace and autocorrelation of the variance of model shows that the sampling algorithm is well mixed. Figure 3 shows the regression coefficients are found with low error.

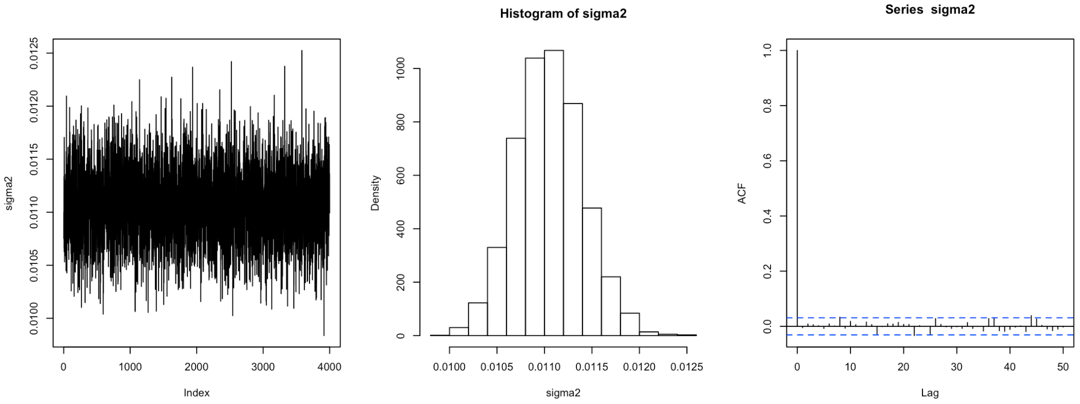

Figure 2. Trace, histogram, and autocorrelation of variance for single component dataset.

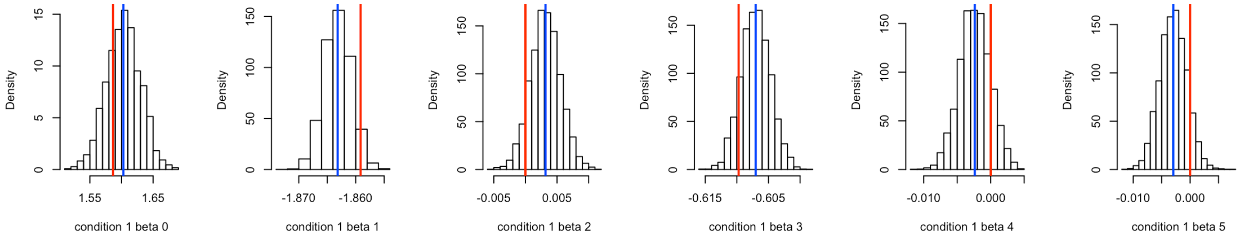

Figure 3. When only one mixing component is present, the regression coefficients are found with small error. Red line is ground truth, blue line is sample mean.

Now we simulate a dataset with 2 mixing components and fit our model. The results are shown in the following figures. We are still able to identify most of the parameters with small error. It is noteworthy that the first component gives misleading result: TF5 (beta 5) is not supposed to be a regulator, but we may conclude that it is a regulator given beta 5 distribution.

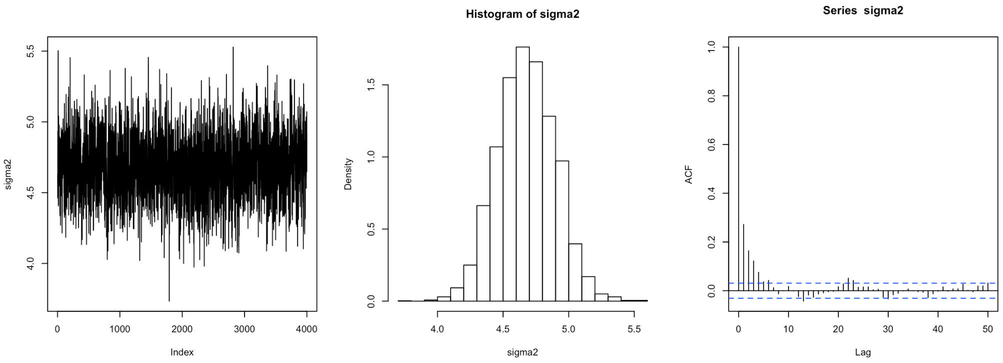

Figure 4. Trace, histogram, and autocorrelation of variance for two components dataset.

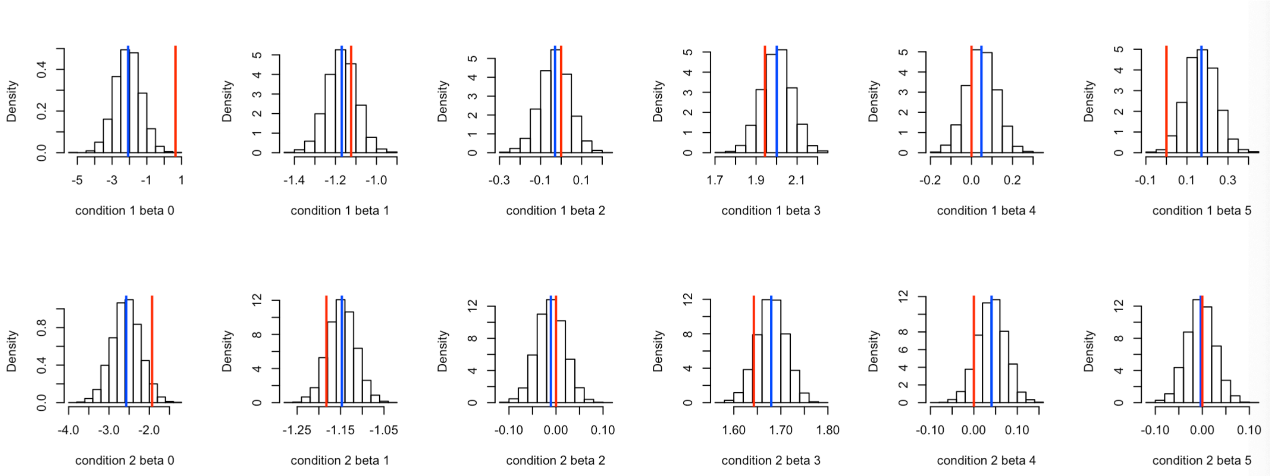

Figure 5. Regression coefficients when two mixing components are present.

Now if we simulate a dataset with 5 mixing components, we have very interesting observations. First, the sampler is very confused and the trace seems to be jumping in different modes. Second, we are still able to identify TFs that are not true regulators, i.e. TF2, TF4 and TF5. But identifying true regulators regression coefficients (beta 1 and beta 3) seem challenging. Third, we see bimodal sampling result in some of the regression coefficients, such as condition 1 beta 4.

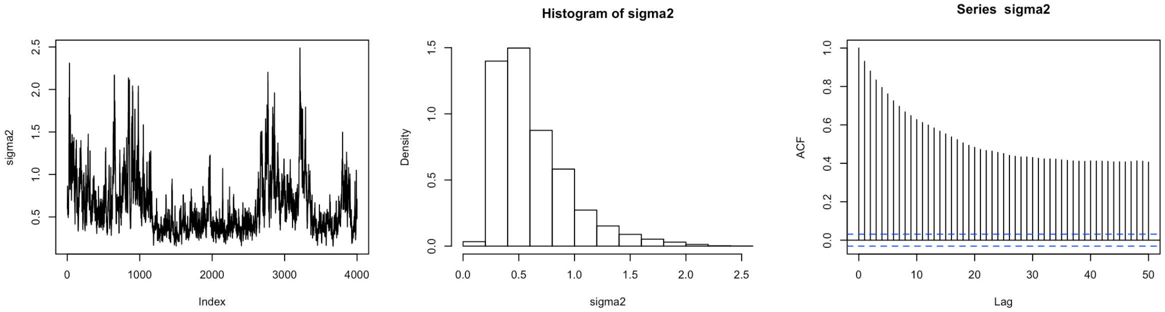

Figure 6. A confused sampler.

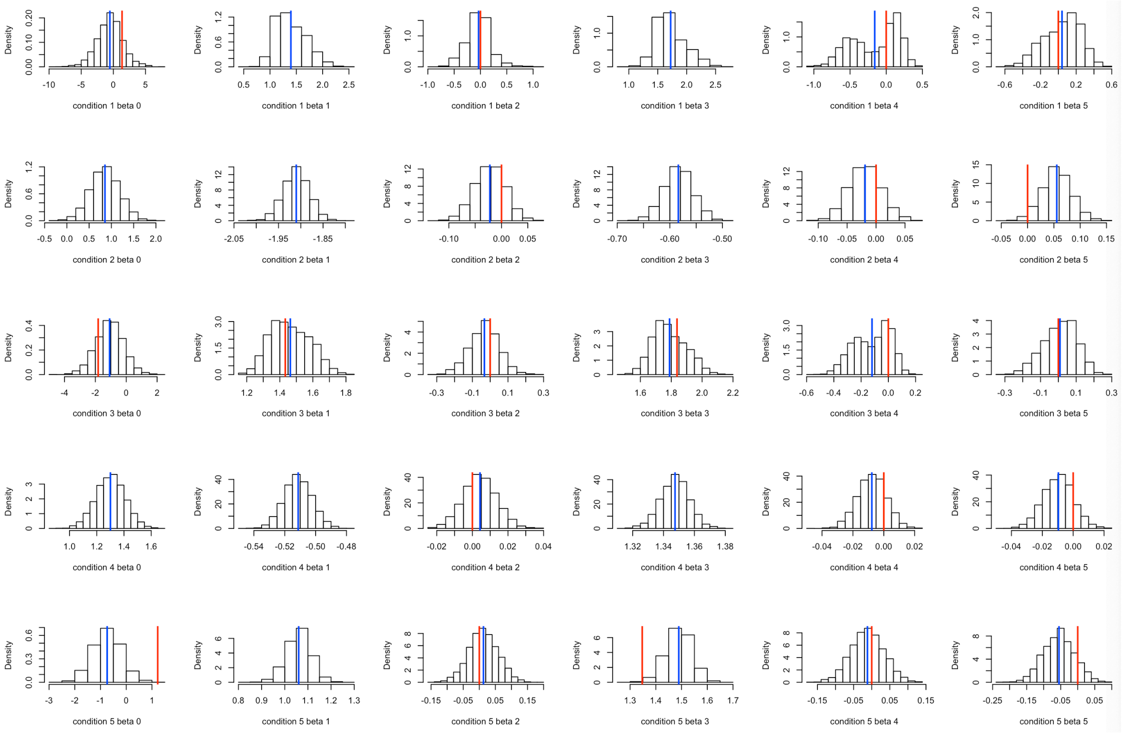

Figure 7. Regression coefficients when five mixing components are present.

In addition to unsatisfactory sampler performance, another issue associated with large number of mixing components is that, the chance that some mixing component NOT being sampled increases. Recall in our sampling algorithm, we associate each data sample with a component by sampling e_i for i=1, 2, …, N. When there is a large number of mixing components, it becomes likely that some components are NOT sampled for any of the N samples. But our priors are sampled assuming all mixing components are present (recall we sample parameters with k=1, 2, …, K, without checking if any k is NOT present in likelihood). If this happens, the posterior estimate will be inaccurate.

This problem may be mitigated by marginalizing the latent variable ($\boldsymbol{e}$) from the joint posterior before inference, i.e.

$$P(\boldsymbol{\theta}, \boldsymbol{\lambda} | \boldsymbol{y}) = \sum_k P(\boldsymbol{\theta}, \boldsymbol{e}, \boldsymbol{\lambda} | \boldsymbol{y}) = P(\boldsymbol{\lambda}) \prod_i \sum_k P(y_i | \boldsymbol{\theta_k}) P(\boldsymbol{\theta_k})\, P(e_i = k | \boldsymbol{\lambda})$$

That is, for each sample i, we find the probability of it generated from each of the k components $P(e_i = k|\boldsymbol{\lambda})$, and use $P(e_i = k|\boldsymbol{\lambda})$ as weights to do a weighted sum of the likelihood of the sample. This is reminiscent of the classic Gaussian Mixture Model, which can be solved using Expectation Maximization, except here instead of maximizing the likelihood, we are interested in maximizing the posterior.

Perhaps the key limitation of this model (an example of finite mixture model) is the hypothesis of data being generated from k components without empirical evidence. Indeed, more complicated modeling scheme such as Dirichlet process mixture modeling which does not assume the number of cluster components a priori are often used to overcome this problem. This will the focus of our next report.

Appendix: Derivation of Full Conditional Distributions

Before deriving the full conditionals, let’s specify the model in full.

The joint distribution for the model is:

$$P(\boldsymbol{\theta}, \boldsymbol{e}, \boldsymbol{\lambda} | \boldsymbol{y}) = P(\boldsymbol{\lambda}) P(\boldsymbol{e} | \boldsymbol{\lambda}) \prod_k P(\boldsymbol{y}_{\boldsymbol{i}:\boldsymbol{e}_{\boldsymbol{i}}=\boldsymbol{k}} | \boldsymbol{\theta}_{\boldsymbol{k}}) P(\boldsymbol{\theta}_{\boldsymbol{k}})$$

The prior for $\boldsymbol{\lambda}$ and $\boldsymbol{e}$:

$$P(\boldsymbol{e}, \boldsymbol{\lambda}) \propto P(\boldsymbol{\lambda}) P(\boldsymbol{e} | \boldsymbol{\lambda}) \propto \prod_k \lambda_k^{0.5 + |i:e_i=k|}$$

The prior for data generated under k^th condition:

$$\begin{aligned} P(\boldsymbol{\theta}_{\boldsymbol{k}}) &\propto P(c_k) P(\boldsymbol{r}_{\boldsymbol{k}} | \boldsymbol{\gamma}_{\boldsymbol{k}}) P(\boldsymbol{\gamma}_{\boldsymbol{k}}) P(\boldsymbol{\beta}_{\boldsymbol{k}} | \boldsymbol{r}_{\boldsymbol{k}}, c_k) P(\sigma_k^2) \\ &\propto c_k^{-g_1-1} \exp\left(-\frac{g_2}{c_k}\right) \cdot \prod_{\boldsymbol{j}} \gamma_{kj}^{r_{kj}+a_j-1} (1-\gamma_{kj})^{b_j - r_{kj}} \\ &\cdot \left(c_k^{J-\sum_j r_{kj}} (s^2)^{J+1}\right)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2} \boldsymbol{\beta}_{\boldsymbol{k}}^{\boldsymbol{T}} \left(\left[1, c_k^{1-r_{k1}}, \dots, c_k^{1-r_{kJ}}\right] \circ s^2 \boldsymbol{I}_{\boldsymbol{J}+\boldsymbol{1}}\right)^{-1} \boldsymbol{\beta}_{\boldsymbol{k}}\right\} \\ &\cdot \left(\frac{1}{\sigma_k^2}\right)^{m_1+1} \exp\left\{-\frac{m_2}{\sigma_k^2}\right\} \end{aligned}$$

The likelihood for data generated under k^th condition

$$P\left(\boldsymbol{y}_{\boldsymbol{i}:e_i=k} \middle| \boldsymbol{\beta}_{\boldsymbol{k}}, \sigma_k^2, \boldsymbol{e}\right) = N\left(\boldsymbol{y}_{\boldsymbol{i}:e_i=k} | \boldsymbol{X}_{\boldsymbol{i}:e_i=k} \boldsymbol{\beta}_{\boldsymbol{k}}, \sigma_k^2 \boldsymbol{I}_{|i:e_i=k|}\right) \propto (\sigma_k^2)^{-\frac{|i:e_i=k|}{2}} \exp\left\{-\frac{\left|\boldsymbol{y}_{\boldsymbol{i}:e_i=k} - \boldsymbol{X}_{\boldsymbol{i}:e_i=k} \boldsymbol{\beta}_{\boldsymbol{k}}\right|_2^2}{2\sigma_k^2}\right\}$$

Now we start deriving the full conditional distributions.

The conditional probability of $\boldsymbol{\lambda}$

$$P(\boldsymbol{\lambda} | \boldsymbol{\theta}, \boldsymbol{e}) \propto \prod_k \lambda_k^{0.5 + |i:e_i=k|}$$

Therefore

$$\boldsymbol{\lambda} | \boldsymbol{\theta}, \boldsymbol{e} \sim Dirichlet(1.5 + |i: e_i = 1|, \dots, 1.5 + |i: e_i = K|)$$

The conditional probability of e_i, i.e. the i^th sample’s condition

$$P(e_i|\boldsymbol{\lambda},\boldsymbol{\theta},y_i) \propto \prod_k \left[\lambda_k^{I_{e_i=k}}\left(\sigma_k^2\right)^{-\frac{I_{e_i=k}}{2}} \exp\left\{-\frac{|y_i-\boldsymbol{X_i\beta_k}|_2{}^2}{2\sigma_k^2}\right\}\right]$$

Therefore

$$e_i|\boldsymbol{\lambda},\boldsymbol{\theta},y_i \sim Categorical\left(\lambda_1(\sigma_1^2)^{-\frac{1}{2}}\exp\left\{-\frac{(y_i-\boldsymbol{X_i\beta_1})^2}{2\sigma_1^2}\right\},\dots,\lambda_K(\sigma_K^2)^{-\frac{1}{2}}\exp\left\{-\frac{(y_i-\boldsymbol{X_i\beta_K})^2}{2\sigma_K^2}\right\}\right)$$

The rest parameters are dependent on condition e. To keep notation uncluttered, we use subscript k to denote that the parameter is on condition of e=k.

The conditional probability for $\boldsymbol{\gamma_k}$:

$$\boldsymbol{\gamma_k}|\boldsymbol{\theta}_{-\boldsymbol{\gamma_k}} \sim \prod_j Beta\left(\gamma_{kj}|a_j+r_{kj},b_j-r_{kj}+1\right)$$

The conditional probability for r_{kj}:

$$P\left(r_{kj}\middle|\boldsymbol{\theta}_{-r_{kj}}\right) \propto \gamma_{kj}^{r_{kj}+a_j-1}\left(1-\gamma_{kj}\right)^{b_j-r_{kj}}\left(c_k^{1-r_{kj}}\right)^{-\frac{1}{2}}\exp\left\{-\frac{\beta_{kj}^2}{2c_k^{1-r_{kj}}s^2}\right\}$$

It does not have a familiar form to some common distributions.

The conditional probability for c_k:

$$\begin{aligned}P\left(c_k\middle|\boldsymbol{\theta}_{-c_k}\right) &\propto c_k^{-g_1-1}\exp\left(-\frac{g_2}{c_k}\right)\prod_j\left[\left(c_k^{1-r_{kj}}s^2\right)^{-\frac{1}{2}}\exp\left(-\frac{\beta_{kj}^2}{2c_k^{1-r_{kj}}s^2}\right)\right]\\ &\propto c_k^{-g_1-1}\exp\left(-\frac{g_2}{c_k}\right)\cdot\left(c_k^{J-\sum_j r_{kj}}\right)^{-\frac{1}{2}}\exp\left(-\frac{\sum_{j:r_{kj}=0}\beta_{kj}^2}{2c_k s^2}\right)\end{aligned}$$

Therefore

$$c_k|\boldsymbol{\theta}_{-c_k} \sim IG\left(c_k\middle|g_1+\frac{J-\sum_j r_{kj}}{2},g_2+\frac{\sum_{j:r_{kj}=0}\beta_{kj}^2}{2s^2}\right)$$

The conditional probability for $\boldsymbol{\beta_k}$:

$$P\left(\boldsymbol{\beta_k}\middle|\boldsymbol{\theta}_{-\boldsymbol{\beta_k}}\right) \propto \exp\left\{-\frac{1}{2}\beta_k^T\left(c_k^{[0,\mathbf{1}-\,\boldsymbol{r_k}]}s^2\boldsymbol{I_{J+1}}\right)^{-1}\beta_k\right\}\exp\left\{-\frac{\left|\boldsymbol{y}_{\boldsymbol{i}:e_i=k}-\boldsymbol{X}_{\boldsymbol{i}:e_i=k}\boldsymbol{\beta_k}\right|_2{}^2}{2\sigma_k^2}\right\}$$

Denote $\left[1, c_k^{1-r_{k1}}, \dots, c_k^{1-r_{kJ}}\right] \circ s^2\boldsymbol{I_{J+1}} = \boldsymbol{\Sigma}, \boldsymbol{y}_{\boldsymbol{i}:e_i=k} = \boldsymbol{y}, \boldsymbol{X}_{\boldsymbol{i}:e_i=k} = \boldsymbol{X}$

$$P\left(\boldsymbol{\beta_k}\middle|\boldsymbol{\theta}_{-\boldsymbol{\beta_k}}\right) \propto \exp\left\{-\frac{1}{2}\boldsymbol{\beta_k^T\Sigma^{-1}\beta_k}\right\}\exp\left\{-\frac{(\boldsymbol{y}-\boldsymbol{X\beta_k})^{\boldsymbol{T}}(\boldsymbol{y}-\boldsymbol{X\beta_k})}{2\sigma_k^2}\right\}$$

$$\propto \exp\left(-\frac{\boldsymbol{\beta_k^T}\boldsymbol{\varSigma^{-1}}\boldsymbol{\beta_k}-\frac{2}{\sigma_k^2}\boldsymbol{y^T X\beta_k}+\boldsymbol{\beta_k^T}\frac{\boldsymbol{X^T X}}{\sigma_k^2}\boldsymbol{\beta_k}}{2}\right)$$

$$\propto \exp\left(-\frac{\boldsymbol{\beta_k^T}\left(\boldsymbol{\varSigma^{-1}}+\frac{\boldsymbol{X^T X}}{\sigma_k^2}\right)\boldsymbol{\beta_k}-\frac{2}{\sigma_k^2}\boldsymbol{y^T X\beta_k}}{2}\right)$$

We can complete the square

$$P\left(\boldsymbol{\beta_k}\middle|\boldsymbol{\theta}_{-\boldsymbol{\beta_k}}\right) \propto \exp\left(-\frac{(\boldsymbol{\beta_k}-\boldsymbol{\mu})^{\boldsymbol{T}}\boldsymbol{Q^{-1}}(\boldsymbol{\beta_k}-\boldsymbol{\mu})}{2}\right)$$

Where

$$\boldsymbol{Q} = \left(\boldsymbol{\Sigma^{-1}}+\frac{\boldsymbol{X^T X}}{\sigma_k^2}\right)^{\boldsymbol{-1}}$$

$$\boldsymbol{\mu} = \frac{1}{\sigma_k^2}\left(\boldsymbol{\Sigma^{-1}}+\frac{\boldsymbol{X^T X}}{\sigma_k^2}\right)^{\boldsymbol{-1}}\boldsymbol{X^T y}$$

Therefore

$$\boldsymbol{\beta_k}|\boldsymbol{\theta}_{-\boldsymbol{\beta_k}} \sim N\left(\boldsymbol{\beta_k}\middle|\frac{1}{\sigma_k^2}\left(\boldsymbol{\Sigma^{-1}}+\frac{\boldsymbol{X^T X}}{\sigma_k^2}\right)^{\boldsymbol{-1}}\boldsymbol{X^T y}, \left(\boldsymbol{\Sigma^{-1}}+\frac{\boldsymbol{X^T X}}{\sigma_k^2}\right)^{\boldsymbol{-1}}\right)$$

Where $\boldsymbol{\Sigma} = \left[1, c_k^{1-r_{k1}}, \dots, c_k^{1-r_{kJ}}\right] \circ s^2\boldsymbol{I_{J+1}}, \boldsymbol{y} = \boldsymbol{y}_{\boldsymbol{i}:e_i=k}, \boldsymbol{X} = \boldsymbol{X}_{\boldsymbol{i}:e_i=k}$

The conditional probability for σ_k^2

$$\sigma_k^2|\boldsymbol{\theta}_{-\sigma_k^2} \sim IG\left(\sigma_k^2\middle|m_1+\frac{|i: e_i = k|}{2}, m_2+\frac{\left|\boldsymbol{y}_{\boldsymbol{i}:e_i=k}-\boldsymbol{X}_{\boldsymbol{i}:e_i=k}\boldsymbol{\beta_k}\right|_2{}^2}{2}\right)$$