Published in final edited form as:
Comput Stat. 2010 March ; 25(1): 17–38. doi:10.1007/s00180-009-0159-7.

Bayesian model-based tight clustering for time course data

Yongsung Joo,
Department of Statistics, Dongguk University, Seoul 100-715, Korea, yongsungjoo@dongguk.edu

G. Casella, and
Department of Statistics, University of Florida, Gainesville, FL 32611, USA

J. Hobert
Department of Statistics, University of Florida, Gainesville, FL 32611, USA

Abstract

Cluster analysis has been widely used to explore thousands of gene expressions from microarray analysis and identify a small number of similar genes (objects) for further detailed biological investigation. However, most clustering algorithms tend to identify loose clusters with too many genes. In this paper, we propose a Bayesian tight clustering method for time course gene expression data, which selects a small number of closely-related genes and constructs tight clusters only with these closely-related genes.

Keywords

Bayesian cluster analysis; Tight clustering; Time course gene expression; Microarray

1 Introduction

Clustering methods can be categorized into heuristic and model-based frameworks. Methods in heuristic frameworks identify clusters based on non-probabilistic measures. The K-means (Hartigan and Wong 1979) and hierarchical (Johnson and Wichern 2002) algorithms belong to this framework. Methods in model-based frameworks cluster objects based on probabilistic measures. As one of the most popular methods in this framework (Basford and McLachlan 1985; Basford et al. 1997), there is the mixture model

$$f(\mathbf{Y}|\theta)=\prod_{i=1}^{n}\sum_{k=1}^{K}\xi_k f(\mathbf{Y}_i|\theta_k), \tag{1}$$

where $\mathbf{Y}_i$ is the response variable of the ith object, K is the number of components, ξ_k is the mixing probability of component k and $f(\mathbf{Y}_i|\boldsymbol{\theta}_k)$ is the probability density function of component k with parameter θ_k. This model has been studied for microarray data analyses in numerous papers (Ghosh and Chinnaiyan 2002, McLachlan et al. 2002; Datta and Datta 2003; Ouyang et al. 2004). Recently, the product partition model (Crowley 1997) have been proposed as alternative model-based approaches using the cluster likelihood:

Correspondence to: Yongsung Joo.

$$f(\boldsymbol{Y}|\theta,\omega)=\prod_{k=1}^{c(\omega)} f\left(\boldsymbol{Y}_{\mathscr{C}_k}|\theta_k\right) \tag{2}$$

where ω is a fixed unknown partition of n objects, $c(\omega)$ is the number of clusters within ω, $\mathscr{C}_k$ is a set of object indices for cluster k, and $\mathbf{Y}_{\mathscr{C}_k}$ is the data of objects in cluster k. Note that partition ω is a free parameter, which should be estimated, and ω determines $c(\omega)$. Then, $\cup_{k=1}^{c(\omega)}\mathscr{C}_k=\{1,2,\ldots,n\}$, and $\mathscr{C}_i \cap \mathscr{C}_j = \varnothing$ when $i \neq j$. These methods assume that the data vectors are partitioned into $c(\omega)$ clusters according to ω and the clusters are independent of each other. While the mixture model (1) is constructed with a known number of clusters, the cluster likelihood (2) contains a partition ω as a parameter to be estimated. In other words, pre-specification of the number of clusters is not needed in the cluster likelihood approach.

As microarray technology became more easily available, biologists can measure gene expressions consecutively over time and examine temporal changes of these expressions. Naturally, many new statistical methods have been developed to cluster genes based on these temporal changes (profiles) in both heuristic (Peddada et al. 2003; Hakamada et al. 2006; Lukashin and Fuchs 2001) and model-based (Schliep et al. 2003; Luan and Li 2003; James and Sugar 2003; Tseng and Wong 2005; Ma et al. 2006; Leng and Muller 2006; Ng et al. 2006) frameworks as follows.

1. Heuristic framework: Peddada et al. (2003) grouped profiles into, so-called, clusters of inequality profiles. For example, suppose a cluster contains profiles with monotonically increasing temporal trends, which can be characterized with inequalities among means at each time point. Similarly, various types of clusters are pre-specified with inequalities, then profiles are clustered based on bootstrap-based criterion. Hakamada et al. (2006) developed a method that cluster profiles based euclidian distances. Lukashin and Fuchs (2001) applied K-means algorithm to temporal profiles.
2. Model-based framework: Leng and Muller (2006) applied functional discriminant analysis considering profiles as independent realizations of a smooth stochastic process. Luan and Li (2003), James and Sugar (2003), Ng et al. (2006), Ma et al. (2006) developed the mixture of mixed effect models to cluster temporal profiles. In their models, $f(Y_i|\theta_k)$ in (1) is specified with a mixed effect model. Particularly, James and Sugar (2003) emphasized application of their model to sparsely and irregularly measured time course data. The Bayesian objective function approach in Booth et al. (2008) and the hidden Markov model in Schliep et al. (2003) are developed based on the cluster likelihood function (2).
3. Costa et al. (2004), Thalamuthu et al. (2006) and Ma et al. (2006) compared clustering methods for time course data using simulation studies.

As the main example of this paper, we analyze the corneal wound healing data, in which expressions of 646 genes are measured twice at irregular 12 time points, using 24 rats (= 2 replicates × 12 time points). The goal of this paper is to identify clusters that contain a small number of genes with very similar temporal patterns. We consider that two types of gene sets, closely-related and weakly-related genes, are included in a microarray experiment either intentionally or unintentionally. If a gene has a close relationship with any other gene(s), we will call it a closely-related gene. Otherwise, we will call it a weakly-related gene. Usually, among thousands of genes on a microarray slide, a large portion of genes are weakly-related. These weakly-related genes tend to increase noise in search of the optimal partition (or clusters) without providing significant amounts of information (Tseng and

Wong 2005). Therefore, direct application of conventional methods will provide large and loose clusters that consist of both closely- and weakly-related genes. However, this is not a desirable result, because biologists typically want to conduct further biological research on a small number of closely-related genes after gene expressions are explored with microarray analyses. To overcome this problem, Tseng and Wong (2005) proposed so-called *tight* clustering method for *cross-sectional* data. The main idea of Tseng and Wong (2005) is to construct tight clusters only with closely-related genes, which is a small portion of the whole data set. Their algorithm can be summarized as follows:

Step 1. With the given number of clusters κ, apply K-means algorithm to subsets (e.g. 70% of genes) of the original data from resampling.

Step 2. Construct candidate tight clusters with genes tends to be together in these resampled subsets.

Step 3. Apply Step 1 and 2 with different κ in a certain range.

Step 4. Identify the final tight clusters that tend to be stable even when κ changes.

Performance of this tight clustering method (Tseng and Wong 2005) was demonstrated by Thalamuthu et al. (2006). In this paper, we propose a new tight clustering algorithm for *time course* gene expression data. Our tight clustering algorithm selects closely-related genes that have high relevance probabilities and then identifies clusters of only closely-related genes using a Bayesian objective function approach (Booth et al. 2008).

In Sect. 2, the Bayesian model and the objective function (Booth et al. 2008) are described in detail with an example of the corneal wound experiment. In Sects. 3 and 4, we discuss a stochastic search algorithm that maximizes a Bayesian objective function and provides simulation studies on this search algorithm. In Sect. 5, we explain how to calculate relevance probabilities and propose the tight clustering algorithm for time course data. To distinguish from tight clustering, we will call algorithms that do not employ the idea of tight clustering, *plain* clustering methods. In Sect. 6, our tight clustering method is applied to the corneal wound data.

2 Bayesian modelling and objective function

We look at an example of the corneal wound healing data to explain the Bayesian model and the objective function by Booth et al. (2008). In the experiment, 2 replicates of 646 gene expressions were measured at each of 12 time points (day 0, 1, 2, 3, 4, 5, 6, 7, 14, 21, 42 and 98) with a corneal wound. In our cluster analysis, the time variable is relabeled with orders 1, 2, …, 12. Justification of relabeling will be discussed in Sect. 6. Because we are interested in the temporal changes of gene expressions rather magnitudes of expressions, gene profiles are centered at the mean of each profile for our analysis.

Averages of two replicates are calculated for each gene at each time point. Then, average expressional profiles of 646 genes are drawn in Fig. 1. Although the patterns of profiles are not easily distinguishable, it seems that there are at least two clusters with increasing and decreasing gene expression patterns in the right end part of the profiles. Also, temporal patterns are not simple enough to use a parametric regression approach and there does not seem to be any periodic pattern. Therefore, within the clustering model, we use the penalized regression spline to explain the mean temporal trend of gene profiles within each cluster.

Given ω, let $\theta=(\theta_k)_{k=1}^{c(\omega)}$ denote a set of cluster-specific parameter vectors $\boldsymbol{\theta}_k$. Then, the marginal posterior distribution of ω is

$$\begin{aligned}\pi(\omega|\boldsymbol{Y}) &\propto \int_{c(\omega)} f(\boldsymbol{Y}|\theta,\omega)\,\pi(\theta|\omega)\,\pi(\omega)d\theta \\ &= \int \prod_{k=1}^{} \left\{ f(\boldsymbol{Y}_{C_k}|\theta_k)\,\pi(\theta_k|\omega) \right\} \pi(\omega)d\theta \\ &=: \text{Obj}(\omega), \end{aligned} \tag{3}$$

where $f(\mathbf{Y}|\boldsymbol{\theta}, \omega)$ is the sampling distribution of the whole data set, $f(\mathbf{Y}_{\mathscr{C}_k}|\boldsymbol{\theta}_k)$ is the cluster specific sampling distribution, and $\pi(\cdot)$ denotes a prior distribution. In the analysis of corneal wound data, $f(\mathbf{Y}_{\mathscr{C}_k}|\boldsymbol{\theta}_k)$ is set to be the normal probability density function, of which the mean is the cluster-specific penalized regression spline. Booth et al. (2008) proposes obtaining the optimal partition ω^* that maximizes the marginal posterior probability $\pi(\omega|\mathbf{Y})$. Because the normalizing constant of $\pi(\omega|\mathbf{Y})$ is difficult to calculate, Obj(ω) is used as the actual objective function in optimizing $\pi(\omega|\mathbf{Y})$. In Sect. 2.1, $f(\mathbf{Y}_{\mathscr{C}_k}|\boldsymbol{\theta}_k)$ is specified in detail with the penalized regression spline. In Sect. 2.2, $\pi(\boldsymbol{\theta}_k|\omega)$ and $\pi(\omega)$ are specified. Finally, in Sect. 2.3, the Bayesian objective function is calculated for our clustering model.

2.1 Penalized regression spline for profiles within a cluster: $f(\mathbf{Y}_{\mathscr{C}_k}|\boldsymbol{\theta}_k)$

In our clustering model, all profiles in cluster k are assumed to have a common smooth underlying trend and independent and identically distributed normal errors with a common variance. Detailed explanation of modeling profiles within a cluster is given as follows. Denote the gene expressions by

$$\boldsymbol{Y} = \left(\boldsymbol{Y}_1^T, \ldots, \boldsymbol{Y}_i^T, \ldots, \boldsymbol{Y}_n^T \right)^T,$$

where $\boldsymbol{Y}_i = \left(\boldsymbol{Y}_{i1}^T, \ldots, \boldsymbol{Y}_{ij}^T, \ldots, \boldsymbol{Y}_{ir}^T \right)$, $\boldsymbol{Y}_{ij}^T = \left(Y_{ij1}, \ldots, Y_{ijt}, \ldots, Y_{ijp} \right)$, and Y_{ijt} is the expression of gene i in the jth replication at time point t. Similarly, define the time variables $\mathbf{x}_i = \left(\mathbf{x}_{i1}^T, \ldots, \mathbf{x}_{ij}^T, \ldots, \mathbf{x}_{ir}^T \right)$, $\mathbf{x}_{ij}^T = \left(x_{ij1}, \ldots, x_{ijt}, \ldots, x_{ijp} \right)$, and $x_{ijt} = t$, and error terms $\varepsilon_i = \left(\varepsilon_{i1}^T, \ldots, \varepsilon_{ij}^T, \ldots, \varepsilon_{ir}^T \right)$, $\varepsilon_{ij}^T = \left(\varepsilon_{ij1}, \ldots, \varepsilon_{ijt}, \ldots, \varepsilon_{ijp} \right)$ and $\varepsilon_{ijt} \sim N(0, \sigma_k^2)$. To explain a temporal profile in cluster k, we use the penalized regression spline:

$$\begin{aligned}\boldsymbol{Y}_{ij} &= \beta_{0k} + \beta_{1k}\mathbf{x}_{ij} + \ldots + \beta_{qk}\mathbf{x}_{ij}^q + \sum_{l=1}^{L} u_{lk}(\mathbf{x}_{ij} - \tau_l)_+^q + \boldsymbol{\varepsilon}_{ij} \\ &= \mathbf{X}\beta_k + \mathbf{Z}\mathbf{U}_k + \boldsymbol{\varepsilon}_{ij}, \end{aligned} \tag{4}$$

where $\mathbf{X} = \left(1, \mathbf{x}_{ij}, \ldots, \mathbf{x}_{ij}^q \right)$, $\beta_k = (\beta_{0k}, \beta_{1k}, \ldots, \beta_{qk})^T$, $\boldsymbol{Z} = \left(z_{ij1}, \ldots, z_{ijl}, \ldots, z_{ijL} \right)$, $z_{ijl} = (\mathbf{x}_{ij} - \tau_l)_+^2$ with knots τ_l's, $m_+ = \max(0, m)$, $\mathbf{U}_k = (u_{1k}, \ldots, u_{Lk})^T$, $\mathbf{0}_p$ is the column vector with p zeros, $\mathbf{I}_p$ is the $p \times p$ identity matrix, and $\varepsilon_{ij} \sim MVN\left(\mathbf{0}_p, \sigma_k^2 \mathbf{I}_p \right)$. If profiles consist of a short time course data, such as 3 time points, the regression spline is not recommended. Then, the simple linear or quadratic regression had better be used instead of the regression spline. The time variable $\mathbf{x}_{ij}$ is the same for every gene i and replicate j in the corneal wound data. For notational simplicity, assume that $\mathscr{C}_k = \{1, \ldots, n_k\}$. Also, let $\boldsymbol{Y}_{\mathscr{C}_k} = \left(\boldsymbol{Y}_1^T, \ldots, \boldsymbol{Y}_{n_k}^T \right)^T$ $\mathbf{X}_{\mathscr{C}_k} = \mathbf{1}_{n_k r} \otimes \mathbf{X}$ and $\mathbf{Z}_{\mathscr{C}_k} = \mathbf{1}_{n_k r} \otimes \mathbf{Z}$, where $\mathbf{1}_{n_k r}$ is a column vector of ones with $n_k r$ elements. Then, within cluster k, the penalized regression spline method estimates parameters by minimizing

$$\|\boldsymbol{Y}_{C_k} - \mathbf{X}_{C_k}\beta_k - \boldsymbol{Z}_{C_k}\mathbf{U}_k\|^2 + \frac{1}{\lambda^2}\|\mathbf{U}_k\|^2. \tag{5}$$

To implement this penalization in the Bayesian framework, two approaches have been widely used. First, the Bayesian Lasso approach of Tibshirani (1996) uses double exponential priors for $\mathbf{U}_k$, which makes finding the posterior distribution mode equivalent to minimization of (5). Second, Ruppert et al. (2003) suggests using the mixed model, of which BLUP (best linear unbiased prediction) is equivalent to the estimates of the penalized regression spline. We use the second approach in this paper by changing a fixed effect $\mathbf{U}_k$ in (4) to a cluster-specific random effect $\mathbf{U}_k \sim N\left(0, \lambda^2\sigma_k^2\mathbf{I}_L\right)$ where $\mathbf{I}_L$ is the $L \times L$ identity matrix.

For data analysis and simulation studies in this paper, we employ a flexible quadratic regression spline by setting $q = 2$ and $L = p - 2$ (one knot at each interior time point). Because the cluster memberships are unknown, it is difficult to get a good prior information on the mean temporal trend of profiles in each cluster. Therefore, the clustering model should contain a flexible spline function with a large number of knots so that it can explain any temporal trend. Also, to prevent a unnecessary wiggly fit, we constrained the influence of knots with the penalty function (Ruppert et al. 2003).

Let $\mathbf{J}_r = \mathbf{1}_r\mathbf{1}_r^T$. Then, the linear mixed model for cluster k can be expressed as:

$$\begin{aligned}\boldsymbol{Y}_{C_k} &= \mathbf{X}_{C_k}\beta_k + \boldsymbol{Z}_{C_k}\mathbf{U}_k + \boldsymbol{\varepsilon}_{C_k} \\ &= \mathbf{X}_{C_k}\beta_k + \boldsymbol{\varepsilon}'_{C_k}\end{aligned} \tag{6}$$

where

$\boldsymbol{\varepsilon}_{\mathscr{C}_k} = \left(\boldsymbol{\varepsilon}_1^T, \ldots, \boldsymbol{\varepsilon}_{n_k}^T\right)^T \sim N(0, \sigma_k^2\mathbf{I}_{n_k rp}), \boldsymbol{\varepsilon}'_{\mathscr{C}_k} \sim N\left(0, \boldsymbol{\Sigma}_{\mathscr{C}_k}\right), \Sigma_{\mathscr{C}_k} = \sigma_k^2\boldsymbol{\Omega}_{\mathscr{C}_k}$ and $\Omega_{\mathscr{C}_k} = \mathbf{I}_{n_k} \otimes \mathbf{I}_r \otimes \mathbf{I}_p + \mathbf{J}_{n_k} \otimes \mathbf{J}_r \otimes \lambda^2\boldsymbol{Z}_{\mathscr{C}_k}\boldsymbol{Z}_{\mathscr{C}_k}^T$.

Then, within cluster k, the likelihood function is

$$f\left(\boldsymbol{Y}_{C_k}|\beta_k, \sigma_k^2\right) = \int f\left(\boldsymbol{Y}_{C_k}, \mathbf{U}_k|\beta_k, \sigma_k^2\right) d\mathbf{U}_k = \frac{1}{(2\pi)^{n_k pr/2}} \frac{1}{\left|\sum_{C_k}\right|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{Y}_{C_k} - \mathbf{X}_{C_k}\beta_k)^T \sum_{C_k}^{-1}(\boldsymbol{Y}_{C_k} - \mathbf{X}_{C_k}\beta_k)\right\}.$$

2.2 Priors: π(ω) and π (β, σ²|ω)

As for the partition parameter ω, we use Crowley's prior (Crowley 1997):

$$\pi(\omega) \propto \varrho^{c(\omega)} \prod_{k=1}^{c(\omega)} (n_k - 1)!, \tag{7}$$

where n_k is the number of genes in cluster k and $\varrho(> 0)$ is the tuning parameter for the size of clusters. Large values of ϱ makes the prior give high probabilities to partitions with a large number of clusters.

For $\beta = (\beta_k)_{k=1}^{c(\omega)}$ and $\sigma^2(\sigma_k^2)_{k=1}^{c(\omega)}$, we use a non-informative prior,

$$\pi\left(\beta,\sigma^2|\omega\right)=\prod_{k=1}^{c(\omega)}\pi\left(\beta_k,\sigma_k^2|\omega\right)\propto\prod_{k=1}^{c(\omega)}\left(1/\sigma_k^2\right)^{\alpha+1}. \tag{8}$$

2.3 Bayesian objective function: Obj(ω)

The marginal posterior distribution of ω is

$$\begin{aligned}\pi(\omega|\boldsymbol{Y}) &\propto \iint f\left(\boldsymbol{Y}|\beta,\sigma^2,\omega\right)\pi\left(\beta,\sigma^2|\omega\right)\pi(\omega)\,d\beta d\sigma^2\\ &=\pi(\omega)\iint\prod_{k=1}^{c(\omega)}f\left(\boldsymbol{Y}_{C_k}|\beta_k,\sigma_k^2\right)\pi\left(\beta_k,\sigma_k^2|\omega\right)\,d\beta d\sigma^2\\ &=\pi(\omega)\prod_{k=1}^{c(\omega)}\iint f\left(\boldsymbol{Y}_{C_k}|\beta_k,\sigma_k^2\right)\pi\left(\beta_k,\sigma_k^2|\omega\right)d\beta_k d\sigma_k^2.\end{aligned}$$

Also,

$$\begin{aligned}&\iint f\left(\boldsymbol{Y}_{C_k}|\beta_k,\sigma_k^2\right)\pi\left(\beta_k,{\sigma_k}^2|\omega\right)\,d\beta_k d\sigma_k^2\\ &=\iint\frac{1}{(2\pi)^{n_k pr/2}}\frac{1}{|\sum_{C_k}|^{1/2}}\exp\left\{-\frac{1}{2}(\boldsymbol{Y}_{C_k}-\mathbf{X}_{C_k}\beta_k)^T\sum\nolimits_{C_k}^{-1}(\boldsymbol{Y}_{C_k}-\mathbf{X}_{C_k}\beta_k)\right\}\times\left(1/\sigma_k^2\right)^{\alpha+1}d\beta_k d\sigma_k^2\\ &=\frac{2^{\alpha}\pi^{(q+1)/2}\Gamma(\nu_k/2)}{S_k^{\nu_k/2}|\boldsymbol{\Omega}_{C_k}|^{1/2}|\mathbf{X}_{C_k}^T\boldsymbol{\Omega}_{C_k}^{-1}\mathbf{X}_{C_k}|^{1/2}},\end{aligned}$$

where

$\nu_k=n_kp-q-1+2\alpha$, $S_k=\left(\boldsymbol{Y}_{\mathscr{C}_k}-\mathbf{X}_{\mathscr{C}_k}\widehat{\beta}_k\right)^T\boldsymbol{\Omega}_{\mathscr{C}_k}^{-1}\left(\boldsymbol{Y}_{\mathscr{C}_k}-\mathbf{X}_{\mathscr{C}_k}\widehat{\beta}_k\right)$, and $\widehat{\beta}_k=\left(\mathbf{X}_{C_k}^T\Omega_{\mathscr{C}_k}^{-1}\mathbf{X}_{\mathscr{C}_k}\right)^{-1}\left(\mathbf{X}_{\mathscr{C}_k}^T\boldsymbol{\Omega}_{\mathscr{C}_k}^{-1}\boldsymbol{Y}_{\mathscr{C}_k}\right)$. Thus,

$$\begin{aligned}\pi(\omega|\boldsymbol{Y}) &\propto \varrho^{c(\omega)}\prod_{k=1}^{c(\omega)}\frac{(n_k-1)!2^{\alpha}\pi^{(q+1)/2}\Gamma(\nu_k/2)}{S_k^{\nu_k/2}|\boldsymbol{\Omega}_{C_k}|^{1/2}|\mathbf{X}_{C_k}^T\boldsymbol{\Omega}_{C_k}^{-1}\mathbf{X}_{C_k}|^{1/2}}\\ &=:\ \mathrm{Obj}(\omega)\end{aligned} \tag{9}$$

Note that the marginal posterior distribution (9) varies with the scale of **Y**. For example, when the scale of **Y** changes to $a\mathbf{Y}$

$$\pi(\omega|a\boldsymbol{Y})\propto\pi(\omega|\boldsymbol{Y})a^{c(\omega)((q+1)-2\alpha)}.$$

Therefore, to make (9) invariant to the scale of **Y**, we set $\alpha=(q+1)/2$.

3 Stochastic search for the optimal partition

Because ω is not a continuous variable, many popular numeric search algorithms, i.e. the Newton–Rapson method, may not be applied. Also, the space of ω is very large, because the Bell number (B_n, the number of all possible partitions) grows rapidly with n (super-exponentially). For example, when n is 10, the Bell number is approximately 10^5. When n is 75, the Bell number becomes approximately 10^{78}, which is a rough estimate for the number

of atoms in the observable universe. The corneal wound data has $n = 646$ genes, which, in a practical sense, have an infinite number of possible partitions. Therefore, searching for the optimal partition ω* is a challenging problem.

To search for the optimal partition, Booth et al. (2008) proposed to generate random partitions from the posterior function $\pi(\omega|\mathbf{Y})$, which is proportional to our objective function Obj(ω), and select a partition that has the highest Obj(ω). This algorithm tends to generate many partitions that are close to the mode of the posterior distribution, which makes the search algorithm efficient. This is so-called Markov Chain Monte Carlo (MCMC) optimization (Jerrum and Sinclair 1996). If the number of objects is not large or only a small number of partitions have high posterior probability $\pi(\omega|\mathbf{Y})$, then the optimal partition can be easily found. Otherwise, the algorithm may work slowly or may find only a suboptimal partition within a reasonable time. However, according to our experience, these suboptimal partitions are close enough to the optimal partition in practical applications.

Here is a detailed description on the MCMC optimization algorithm (Booth et al. 2008). Within the optimization algorithm, a biased random walk (Metropolis-Hastings algorithm) generates the targeting Markov chain of random partitions from $\pi(\omega|\mathbf{Y})$. Suppose that the biased random walk is iterated R times. Let ω_i be the partition in the i^{th} iteration, $c(\omega_i)$ be the number of clusters in ω_i, $\tilde{\omega}_i^*$ be the partition with the highest value of the objective function during the first i iterations, and $\tilde{\omega}^* \left(=\tilde{\omega}_R^*\right)$ be the best partition that is found by an application of the optimization algorithm. Then, consider the following algorithm:

Algorithm 1

Step 1. Choose an initial partition ω_1 and set $\tilde{\omega}_1^*=\omega_1$ and $i = 1$.

Step 2. Generate a candidate partition ω′:

- If $c(\omega_i)=1$, select one uniform random object out of n and make it as a new singleton cluster (one profile in one cluster). This makes two clusters with 1 and $n - 1$ objects.
- If $c(\omega_i) \geq 2$, select one uniform random object.
- If the selected object is a singleton cluster, move it to one of $c(\omega_i) - 1$ clusters with probability $1/(c(\omega_i) - 1)$.
- Otherwise, move it to one of other clusters with probability $1/c(\omega_i)$ or make its own singleton cluster with probability $1/c(\omega_i)$.

Step 3. Accept ω′ with probability $min(1, \pi\left(\omega_i'\right)/\pi(\omega))$. If accepted, $\omega_{i+1} = \omega'$. Otherwise, $\omega_{i+1} = \omega_i$.

Step 4. If $\text{Obj}(\omega_{i+1})>\text{Obj}\ (\tilde{\omega}_i^*)$, $\tilde{\omega}_{i+1}^*=\omega_{i+1}$. Otherwise, $\tilde{\omega}_{i+1}^*=\tilde{\omega}_i^*$. Set $i = i + 1$ and repeat Steps 2, 3 and 4.

Note that, if R is not large enough, $\tilde{\omega}$* may not be the same as the optimal partition ω*. In this paper, Algorithm 1 is applied three times for each data set of interest. If all chains provide the same $\tilde{\omega}$*, then ω* is considered as ω*.

4 Simulation studies on stochastic search algorithm

Simulation studies are conducted with the corneal wound data, to check the convergence-in-distribution of the biased random walk in Algorithm 1 and to examine the needed number of iterations before Algorithm 1 finds the optimal partition.

4.1 Convergence of the biased random walk

To examine the convergence or performance of the biased random walk algorithm, it is necessary to start chains with fair and non-informative initial partitions. As one of the most reasonable initial partitions, we may consider a uniform random partition that is drawn randomly with a probability of $1/B_n$. However, the generation of a uniform random partition is a challenging problem when the partition space is very large. In this subsection, we describe an algorithm to generate uniform random partitions with a large n, and demonstrate the convergence of chains using the corneal wound data.

Generation of uniform random partitions with a large n—Let $\mathbb{P}_n$ be the set of all possible partitions of $\mathbb{N}_n := \{1, \ldots, n\}$. Also, let Π denote a uniform random partition such that $P(\Pi = \pi) = 1/B_n$ for all $\pi \in \mathbb{P}_n$, so Π has the uniform distribution on $\mathbb{P}_n$. Here we describe a method of simulating a random uniform partition Π.

Let M be a random variable on the set $\{1, 2,\ldots, n\}$ with probabilities given by

$$P(M=m)=\frac{m^n}{B_n}\left[\frac{1}{m!}\sum_{s=0}^{n-m}\frac{(-1)^s}{s!}\right] \tag{10}$$

for $m = 1, 2,\ldots, n$. Pitman (1997) gave an algorithm for drawing a value of Π, which goes as follows. First, draw a value of $M = m$ from (10), then randomly distribute n balls with labels $\mathbb{N}_n$ into the m different urns. Note that some of the urns may end up empty. After excluding empty urn(s), the resulting partition has the uniform distribution on $\mathbb{P}_n$.

Unfortunately, this method does not work well with large n because computation of the distribution (10) requires the evaluation of large factorials, which results in numerical difficulties. However, it is possible to circumvent this problem by approximating (10) to a high degree of accuracy and drawing M from this approximation. Based on this this idea, we propose a new algorithm as follows (See the Appendix for detailed calculation of this algorithm).

Algorithm 2:

Step 1. Choose $\varepsilon > 0$, which controls the degree of approximation. Here we use $\varepsilon = 10^{-30}$

Step 2. Calculate $l_m = n \log m - \log \Gamma(m + 1)$ for $m \in \{1, 2,\ldots, n\}$ and set

$$l^*=\max\{l_1, l_2, \ldots, l_n\}.$$

Step 3. Find

$$N=\inf\left\{m>n: l^* - l_m > \log\left(2^{n+1}n!/\varepsilon\right)\right\}.$$

Step 4. Draw M^* with probabilities given by

$$P(M^*=m)=\frac{\exp\{l_m\}}{\sum_{j=1}^{N}\exp\{l_j\}}=\frac{\exp\{l_m - l^*\}}{\sum_{j=1}^{N}\exp\{l_j - l^*\}} \tag{11}$$

for $m \in \{1, 2, \ldots N\}$.

Uniform random partitions are generated with $n = 646$ as in the corneal wound data. Then, the number of clusters in each partition is counted to draw the histogram in Fig. 2. Most uniform random partitions have from 120 to 143 clusters with mean 131.3 and variance 4.6^2. The gray line in Fig. 2 shows the distribution of the number of bins, m in (11), which has mean 132.3 and variance 4.7^2.

Convergence-in-distribution with the corneal wound data—When multiple chains are simulated with a good MCMC algorithm, they get mixed well regardless of the initial partitions. Using the biased random walk, five chains are started from four non-informative partitions (two uniform random partitions, one partition with $n = 646$ singleton clusters and one partition that has only one cluster for all objects) and one informative partition (with 200 clusters found using the K-means algorithm).

The Bayesian objective function (9) has two parameters, λ and ϱ, that should be set before Algorithm 1 starts. To estimate λ, which is the smoothing parameter or the ratio of two variances in the linear mixed model (6), we clustered profiles into an arbitrary number of groups, $K = 40$, with K-means algorithm, and fitted the linear mixed model to the grouped data assuming a homogeneous error variance across clusters. From this procedure, we got an estimate $\hat{\lambda} = 1.31$. Even though $\hat{\lambda}$ depends on K, $\hat{\lambda}$ is robust to K as long as K is not close to 1 or n. For example, $\hat{\lambda} = 1.27$ when $K = 20$, and $\hat{\lambda} = 1.33$ when $K = 200$. Also, the resulting optimal partition is usually not sensitive to a small variation of $\hat{\lambda}$, such as ± 0.2. For the tuning parameter in Crowley's prior, $\log(\varrho) = 8$ is chosen. Detailed discussion about the selection of $\log(\varrho)$ is in Sect. 6.

Simulation histories of the five chains are shown with the number of clusters on Fig. 3a, and with the objective function $\text{Obj}(\omega)$ on Fig. 3b. Because most uniform random partitions have a similar number of clusters, around 131, and similar values of $\text{Obj}(\omega_1) \approx -2{,}300$, two initial partitions are overlapped on Fig. 3(a, b). Note that uniform random partitions have very low initial values of the objective function, which indicates that they are very non-informative partitions. On the contrary, the initial partition from K-means algorithm has the highest objective function value, $\text{Obj}(\omega_1) = 4{,}597$. Even when five chains have very different initial partitions, Fig. 3 shows that the biased random walk results in excellent agreement at convergence, and good mixing over the stationary distribution after 10^6 iterations. Therefore, we consider that 2×10^6 iterations are enough to be a burn-in period in analyzing the corneal wound data.

The convergence-in-distribution does not guarantee that Algorithm 1 can find the optimal partition ω^* within a reasonable time. For example, in Table 1, the algorithm ($R = 3 \times 10^6$) is applied to the corneal wound data twice with $\log(\varrho) = 0, 3, 5, 8, 10, 13, 15$ and 20. Regardless of the value of $\log(\varrho)$, two chains provide different $\tilde{\omega}^*$'s, which have different $\text{Obj}(\tilde{\omega}^*)$'s and different numbers of clusters. However, two $\text{Obj}(\tilde{\omega}^*)$ are very close to each other, considering that this search is conducted in almost infinite space of a discrete parameter. More details of Table 1 will be discussed in Sect. 6.

4.2 Number of iterations before the optimal partition is found

To examine the relationship between the number (n) of gene profiles to be clustered and the number (R^*) of iterations before the optimal partition ω^* is found by Algorithm 1, data sets are simulated by randomly selecting n gene profiles from the corneal wound data without replacements. Then, three long chains with 5×10^7 iterations are simulated for each data set using the biased random walk. If these three chains have the same $\tilde{\omega}^*$, we consider that the optimal partition $\omega^* = \tilde{\omega}^*$ is found for a simulated data set. The average of three R^*'s,

Avg(*R**), is reported in Table 2. For example, for the first simulated data set with randomly selected n =20 profiles, the same $\tilde{\omega}^*$ is found by three chains after 6,220 iterations on average. In total, 5 data sets are simulated with n = 20. The other four data sets have *Avg*(*R**) = 1,637, 663, 4,439 and 934. All *Avg*(*R**)'s are less than 10^4 when n = 20. When n increases to 50, *Avg*(*R**)'s have magnitude 10^6 or 10^7 with the first four simulated data sets. With the fifth simulated data set, the same optimal partition is not found by three chains, which indicates that more than 5×10^7 iterations are needed. When n = 100, the optimal partition is not found with any data set within 5×10^7 iterations. It seems that *Avg*(*R**) increases super exponentially as the Bell number does ($B_{n=100} \approx 10^{115}$). Our C program spent about 8 h to iterate 5×10^7 times for each data set with n = 100. Therefore, it is practically impossible to find the optimal partition of 646 gene profiles within a reasonable time. There can be two major reasons that make the optimization algorithm slow. First, the discrete space of partitions is too large with $n \geq 100$. Second, there seems to be many suboptimal partitions, of which Obj(ω)'s are close to Obj(ω*). This may happen when many weakly-related genes are included in the data set and cause loose clusters. Also, weakly-related genes have a tendency to change cluster memberships easily during MCMC iterations, because they don't make significant contributions to the objective function. In the next section, we propose a tight clustering algorithm that selects a small number of closely-related genes and applies Algorithm 1 to only these genes.

5 Tight clustering algorithm

Recall that, during the biased random walk of Algorithm 1, a candidate partition ω' is generated by moving one object (gene) in the current partition ω_i. If a gene is closely-related with other genes in the current cluster and build a tight cluster, then this gene have low chances to move during iterations. We will consider a gene to be stable, when a gene doesn't move at all or change only a small number of times over MCMC iterations of Algorithm 1.

Let the relevance probability (RP) of a pair of genes be the probability of an event that two genes in a pair belong to one cluster together in a random partition. In the Bayesian objective function approach framework, RP can be estimated easily by counting how often two genes are together in a chain of the biased random walk. Because closely-related genes tend to stay together within a cluster over the course of MCMC iterations, they have high RP's. Our definition of relevance probability is slightly different from the original concept by Hartigan (1991) where the relevance probability is the probability having a set $\mathbb{S}$of genes (objects) as a cluster in a random partition. It is inappropriate to use Hartigan's definition of RP for finding closely-related genes because there can be too many possible clusters, $\mathbb{S}$ For example, when Hartigan's definition of RP is used to find closely-related genes in corneal wound data, RPs of $2^{646} - 2 \approx 2.9 \times 10^{194}$ (the number of all possible non-empty subsets) possible clusters must be calculated for a simulation chain. When our definition of RP is used, RPs of only $\binom{646}{2} = 208,335$ pairs must be calculated.

Here, we propose the following tight clustering algorithm:

Algorithm 3

Step 1. Select $\log(\varrho^*)$ that makes a small number of genes stable in a chain of the biased random walk.

Step 2. Apply Algorithm 1 with $\log(\varrho^*)$ and estimate RP's of all possible pairs.

Step 3. Apply Algorithm 1 with only closely-related genes (RP$\geq \eta$) to construct tight clusters.

In Step 1, we suggest selecting a value of log (ϱ*) that makes a small number of genes stable because this makes a small number of genes have high RP's in Step 2. However, an analytic optimization for log (ϱ*) is very difficult. Therefore, we suggest running Algorithm 1 with different log(ϱ) values to calculate the number of stable genes. The cutoff value η of RP in Step 3 can be considered as a tuning parameter that determines the tightness of clusters. If η is close to 1, Algorithm 3 will select a small number of genes that construct very tight clusters. If η is close to 0, Algorithm 3 will select most genes without constructing tight clusters. Based on our experience with real data and simulation studies, we suggest setting η = 0.80 for reasonably tight clusters.

To check performance of the tight clustering algorithm (Algorithm 3), we conducted simulation studies with the following six true clusters:

$$\begin{aligned}
C_1&: y_i = \sin(x_i/2) \times \log(x_i)^2 + \varepsilon_{1i}\\
C_2&: y_i = -1 + \exp(x_i/5) \times \cos((x_i - 2)/2)^2 + \varepsilon_{2i}\\
C_3&: y_i = 2 - x_i + \exp(x_i/5) \times \cos(x_i/2)^2 + \varepsilon_{3i}\\
C_4&: y_i = 1 + 1.5 \times \sin(x_i/2) + \varepsilon_{4i}\\
C_5&: y_i = \exp(x_i/5) + \varepsilon_{5i}\\
C_6&: y_i = 3 - x_i + \cos(x_i/2)^2 + \varepsilon_{6i},
\end{aligned}$$

where ε_{1i}, ε_{2i}, ε_{3i}, ε_{4i} and ε_{5i} have independent and identical normal distributions with mean 0 and variance σ_A^2, ε_{6i} has the independent and identical normal distribution with mean 0 and variance σ_B^2, and $x_i = 1, 2, \ldots, 6$. Four sets of simulations are considered with different σ_A and σ_B. In each set, we simulated 60 profiles (10 profiles from each cluster) 100 times. Then, our tight clustering and plain Bayesian objective function approaches (Booth et al. 2008) with various tuning parameter log(ϱ)'s are compared in Table 3. In Step 1 of the tight clustering algorithm, we chose log(ϱ*) for each simulated data set by examining the number of stable genes when log(ϱ) = −10, −8, −6, −4, −2 and 0. In general, log(ϱ) is a very important parameter in the clustering algorithm, controlling the number of clusters. However, when two values of log(ϱ) differed by less than 2 in our simulation experience, the same or very similar clusters were found. In this paper, we will consider a pair of profiles to be correctly clustered if the pair is in the same cluster of the simulation model and is grouped together by the clustering algorithm or if two profiles in the pair are in different clusters of the simulation model and are not grouped together. Otherwise, we will consider the pair to be incorrectly clustered. There are $N_{\text{pairs}} = n(n-1)/2$ possible pairs, where n is the number of profiles to be clustered. In the plain clustering, $n = 60$ and $N_{\text{pairs}} = 1{,}770$ for every simulated data set. However, in the tight clustering, n and N_{pairs} varies with simulated data because only closely-related profiles are interested for clustering and the number of closely-related profiles varies with data. We define that the percentage classification error rates of pairs is:

$$ER = 100 \times \frac{\sum_i^{N_{\text{pairs}}} I_i}{N_{\text{pairs}}},$$

where $i = 1, \ldots, N_{\text{pairs}}$ and

$$I_i = \begin{cases} 1, & \text{if } i\text{th pair is incorrectly clustered;} \\ 0, & \text{otherwise.} \end{cases}$$

See Table 3. For the first set of simulations, we set $\sigma_A = \sigma_B = 0.5$. These are relatively small standard deviations that make clustering algorithms find correct clusters easily. Tight clustering method has the lowest error rate of 6.0% with a standard error of 0.8%. Among plain clusterings, the lowest error rate, 8.8%, is gained when $\log(\varrho) = -10$. Recall that a higher ϱ makes the algorithm choose a large number of small clusters. When $\log(\varrho) = 5$, the algorithm provides the same optimal partition of n singleton clusters for all 100 simulated data sets. A simulated data set has 270 (= 45 × 6) pairs that have two profiles from the same true cluster. Therefore, when all profiles are separated as singleton clusters, the error rate becomes 15.3%(= 100 × 270/1,770). When we increase the standard deviations to $\sigma_A = \sigma_B = 0.7$, the tight clustering algorithm achieves the lowest error rate again, 13.7%. When the standard deviations are as large as $\sigma_A = \sigma_B = 1.0$, it is practically impossible to get good clustering results because most parts of the true clusters overlap each other. Therefore, the error rates are high for both tight and plain clustering methods. In this case, the plain clustering method with $\log(\varrho) = 5$ works the best, only because it separates all profiles as singletons and gains an error rate of 15.3%. However, this is not a desirable result because it does not provide any non-singleton cluster. As the last set of simulations, we considered $\sigma_A = 0.7$ and $\sigma_B = 3.0$, which makes cluster $\mathscr{C}_6$ relatively diffused. The tight clustering method worked the best in this case. Compared to simulations with $\sigma_A = \sigma_B = 0.7$, it seems clustering was easier because $\mathscr{C}_6$ with $\sigma_B = 3.0$ is not similar to other clusters with $\sigma_A = 0.7$.

Overall, we found that the tight clustering method has lower error rates compared to plain clustering when true clusters are reasonably separated. This result can be easily expected because tight clustering uses only closely-related profiles (genes) and excludes weakly-related profiles that may cause erroneous clusters.

6 Analysis of the corneal wound data

In the corneal wound experiment, gene expressions are measured at day = 0, 1, 2, 3, 4, 5, 6, 7, 14, 21, 42 and 98 because expressions are expected to change more intensely in the first week and then get stabilized in the later part of experiment. In other words, a less smooth pattern is expected in the first week. When spline knots are located with equal interval on the real time scale (days), this change of smoothness will not be reflected. In this paper, we suggest relabeling time points with 1,…,12 before the cluster analysis. This may not be the optimal solution, but an easy and reasonable solution to the unequal smoothness problem. Here, this argument is illustrated with an example of a gene profile in our data. Using the same smoothing parameter ($\lambda = 1.3$) and knots at all interior time points, the gene profile is fitted on real scale in Fig. 4a and on relabeled equal interval scale in Fig. 4c. Also, to make comparison of these fits easier, Fig. 4a is redrawn with equal interval time scale in Fig. 4(b, c) is redrawn with real time scale in Fig. 4d. When Fig. 4(a, d) (or, equivalently, Fig. 4b, c) is compared, the difference between two spline fits are negligible from 1st (day=0) to 7th time points (day=6). However, the later part of the profile (i.e. from 7th to 12th time points) is overfitted in Fig. 4a, while it is fitted with a proper smooth line in Fig. 4d. Similar patterns are found with other gene profiles. If analyst wants to use the real time scale, this overfitting problem may be fixed by carefully choosing the number and locations of knots.

Now on, we will provide a detailed discussion about application of the tight clustering algorithm to the corneal wound data.

Step 1

Simulation studies are conducted with $\log(\varrho)$ =0,3,5,8,10,13,15 and 20. For consistency of comparisons, the same initial partition with 200 clusters from the K-means algorithm is used for every simulation. Algorithm 1 is applied twice with each $\log(\varrho)$ and the movements of genes are monitored in the last 10^6 iterations out of 3×10^6 total iterations. Table 1 describes

Obj ($\tilde{\omega}$*), c ($\tilde{\omega}$*), and the number of stable genes, which do not move at all during the last 10^6 iterations. For example, from the first run with log(ϱ) = 0, we got Obj ($\tilde{\omega}$*) = 5,280, c ($\tilde{\omega}$*) = 27 and 58 stable genes out of 646. When log(ϱ) gets larger, the number of clusters c ($\tilde{\omega}$*) increases because a larger value of log(ϱ) makes the objective function support a larger number of clusters, and hence smaller numbers of genes per cluster. In Fig. 5, the number of stable genes is plotted with log(ϱ). It shows that a relatively small number of genes are stable (do not switch clusters in the last 10^6 iterations) when log(ϱ) is between 3 and 13. When log(ϱ) is as large as 20, most genes have a strong tendency to stay as singleton clusters during the iterations. This makes many genes appear stable. When log(ϱ) is as small as 0, even weakly-related genes may stay together easily within large clusters. This also causes many genes to appear stable. Figure 5 shows that the number of stable genes is the minimum around log(ϱ) = 8. Therefore, log(ϱ*) = 8 is selected for the corneal wound data.

Step 2 and Step 3

After applying Algorithm 1 to the whole data with 2×10^7 iterations, the 1,169 pairs are found to have RP≥ 0.80. Then, Algorithm 1 is applied 3 times to only the 139 closely-related genes that compose these 1,169 pairs. The same $\tilde{\omega}$* is found easily in all three applications with less than 10^6 iterations. Recall that, when a data set of 100 profiles is simulated in Sect. 4.2, the optimal partition ω* is not found even after 5×10^7 iterations. The convergence is fast with closely-related genes because only a small number of genes are considered in tight clustering and close relationships among selected genes make Obj(ω*) much higher than any Obj(ω), where $\omega \neq \omega^*$. See Fig. 6 for the final result of our analysis. Each cluster has a distinctive pattern and gene profiles build tight clusters. Cluster 3 and 9 seem similar, but scales distinguish them.

7 Concluding remark

A typical purpose of cluster analysis with microarray data is to identify a small number of closely-related genes that biologists can study further in future studies. Also, weakly-related genes make the clustering algorithm work slowly and build large and loose clusters. Therefore, we propose to select closely-related genes using relevance probabilities and get tight clusters only with closely-related genes.

In our tight clustering, the stochastic search algorithm (Algorithm 1) is used three times for different purposes. In the first two applications with all gene profiles, the stochastic algorithm is used for the selection of log(ϱ*) and the calculation of relevance probabilities of all pairs, rather than for the search of the optimal partition. Because the stochastic algorithm is implemented by MCMC simulation, the calculation of relevance probabilities could be done easily. Instead of the Bayesian objective function approach, if the mixture model is used in tight clustering, it will be difficult to calculate relevance probabilities because the mixture model requires prior knowledge on a fixed number of components, K. Even though K can be determined with a model selection criterion (i.e. BIC), it is difficult to measure uncertainties in determining K and take account of it when calculating relevance probabilities.

In addition to closely-related genes, if a small number of weakly-related genes are interesting for biological reasons, then they can also be included in Step 3 of tight clustering algorithm. We expect that inclusion of these weakly-related genes will not make significant impact on the optimal partition of tight clusters, because closely-related genes build a stable structure of clusters.

In this paper, we clustered expressions of 646 genes that are preselected by biologists. However, in general, most microarray experiments generate 10,000–40,000 gene

expressions at once. When the number of genes are such high, computational time can be considered as a limitation of our approaches. For example, if genes are measured twice at 12 time points as in corneal wound data set, it takes about a week with 3.0Hz PC for our tight clustering method to handle 1,000 or less genes (or objects). Therefore, we recommend preselecting gene profiles that vary largely over time, using the one-way ANOVA model with time as the covariate. For example, we may preselect genes that have the 1,000 highest *F*-values. A similar approach was also proposed in Peddada et al. (2003).

Appendix

It turns out that the Pitman's algorithm (Pitman 1997) still works if we take M to be the random variable on the set $\mathbb{N} := \{1, 2, 3, \ldots\}$ with probabilities given by

$$P(M=m)=\frac{m^n e^{-1}}{B_n m!}$$

for $m = 1, 2, 3, \ldots$. We now describe a method for sampling from (an arbitrarily good approximation of) M without having to calculate B_n. Specifically, given $\varepsilon > 0$, we find a positive integer $N = N(\varepsilon)$ such that $P(M \geq N + 1) < \varepsilon$ and we approximate M with a random variable M^* with probabilities given by $P(M^* = m) \propto m^n/m!$ for $m \in \{1, 2, \ldots, N\}$. If we choose ε small enough, there will be no discernible difference between draws from M and draws from M^*.

Fix $N > 1$ and note that

$$\sum_{j=N+1}^{\infty} P(M=j)=\sum_{j=1}^{\infty} P(M=N+j),$$

and

$$\begin{aligned}\frac{P(M=N+j)}{P(M=N)} &= \frac{(N+j)^n}{(N+j)!}\frac{N!}{N^n}\\ &=(1+j/N)^n\frac{N!}{(N+j)!}\\ &=(1+j/N)^n\frac{1}{(N+j)(N+j-1)\cdots(N+1)}\\ &\leq (1+j)^n\frac{1}{N^j}\\ &\leq \frac{(2j)^n}{N^j}.\end{aligned}$$

Therefore,

$$P(M=N+j) \leq \frac{(2j)^n}{N^j} P(M=N),$$

so

$$\begin{aligned}\sum_{j=N+1}^{\infty} P(M=j) &= \sum_{j=1}^{\infty} P(M=N+j) \\ &\le 2^n P(M=N) \sum_{j=1}^{\infty} \frac{j^n}{N^j} \\ &\le 2^n P(M=N) 2 \int_0^{\infty} x^n e^{-x \log N} dx \\ &= \frac{2^{n+1} P(M=N) n!}{(\log N)^{n+1}} \\ &\le 2^{n+1} n! P(M=N).\end{aligned}$$

Thus, if we can find N such that $P(M = N) < \varepsilon/(2^{n+1}n!)$, then

$$P(M \ge N+1) = \sum_{j=N+1}^{\infty} P(M=j) < \varepsilon.$$

It's easy to show that $P(M = m)$ is decreasing for $m > n$. Define $l_m = n \log m - \log \Gamma(m + 1)$ so that $\log P(M = m) = l_m - c$, where $c = 1 + \log B_n$. Now define $l^* = \max\{l_1, l_2, \ldots, l_n\}$ and note that, for any $m \in \mathbb{N}$,

$$P(M=m) = \frac{\exp\{l_m\}}{\sum_{j=1}^{\infty} \exp\{l_j\}} \le \frac{\exp\{l_m\}}{\exp\{l^*\}} = \exp\{l_m - l^*\}.$$

Therefore, if $l^* - l_m > \log(2^{n+1} n!/\varepsilon)$, we have

$$P(M=m) \le \exp\{l_m - l^*\} \le \varepsilon / \left(2^{n+1} n!\right).$$

Define

$$N = \inf\left\{m > n : l^* - l_m > \log\left(2^{n+1} n!/\varepsilon\right)\right\}.$$

We then calculate the probabilities of M^* as

$$P(M^*=m) = \frac{\exp\{l_m\}}{\sum_{j=1}^{N} \exp\{l_j\}} = \frac{\exp\{l_m - l^*\}}{\sum_{j=1}^{N} \exp\{l_j - l^*\}}$$

for $m \in \{1, 2, \ldots, N\}$.

References

Basford KE, McLachlan GJ. Likelihood estimation with normal mixture models. Appl Stat. 1985; 34:282–289.

Basford KE, Greenway DR, McLachlan GJ, Peel D. Standard errors of fitted means under normal mixture models. Comput Stat. 1997; 12:1–17.

Booth J, Casella G, Hobert J. Clustering using objective functions and stochastic search. J R Stat Soc B. 2008; 70(1):119–140.

Costa IG, Carvalho FAT, Souto MCP. Comparative analysis of clustering methods for gene expression time course data. Genet Mol Biol. 2004; 27:623–631.

Crowley EM. Product partition models for normal means. J Am Stat Assoc. 1997; 92:192–198.

Datta S, Datta S. Comparisons and validation of statistical clustering techniques for microarray gene expression data. Bioinformatics. 2003; 19:459–466. [PubMed: 12611800]

Ghosh D, Chinnaiyan AM. Mixture modelling of gene expression data from microarray experiments. Bioinformatics. 2002; 18:275–286. [PubMed: 11847075]

Hakamada K, Okamoto M, Hanai T. Novel technique for preprocessing high dimensional time-course data from DNA microarray: mathematical model-based clustering. Bioinformatics. 2006; 22:843–848. [PubMed: 16434440]

Hartigan JA. Partition models. Commun Stat Theory Methods. 1991; 19:2745–2756.

Hartigan JA, Wong MA. Algorithm AS 136: a K-means clustering algorithm. Appl Stat. 1979; 28:100–108.

James GM, Sugar CA. Clustering for sparsely sampled functional data. J Am Stat Assoc. 2003; 98:397–408.

Jerrum, M.; Sinclair, A. Approximation algorithms for NP-hard problems. Boston: PWS Publishing; 1996. The Markov Chain Monte Carlo method: an approach to approximate counting and integration.

Johnson, RA.; Wichern, DW. Applied multivariate statistical analysis. 5th edn.. Prentice Hall, Upper Saddle River: 2002.

Leng X, Muller H. Classification using functional data analysis for temporal gene expression data. Bioinformatics. 2006; 22:68–76. [PubMed: 16257986]

Luan Y, Li H. Clustering of time-course gene expression data using a mixed-effects model with B-splines. Bioinformatics. 2003; 19:474–482. [PubMed: 12611802]

Lukashin AV, Fuchs R. Analysis of temporal gene expression profiles: clustering by simulated annealing and determining the optimal number of clusters. Bioinformatics. 2001; 17:405–414. [PubMed: 11331234]

Ma P, Castillo-Davis CI, Zhong W, Liu JS. A data-driven clustering method for time course gene expression data. Nucleic Acids Res. 2006; 34:1261–1269. [PubMed: 16510852]

McLachlan, GJ.; Baford, KE. Mixture models: inference and applications to clustering. New York: Marcel Dekker, Inc; 1988.

McLachlan GJ, Bean RW, Peel D. A mixture model-based approach to the clustering of microarray expression data. Bioinformatics. 2002; 18:413–422. [PubMed: 11934740]

Ng SK, McLachlan GJ, Wang K, Jones LB, Ng SW. A mixture model with random-effects components for clustering correlated gene-expression profiles. Bioinformatics. 2006; 22:1745–1752. [PubMed: 16675467]

Ouyang M, Welsh WJ, Georgopoulos P. Gaussian mixture clustering and imputation of microarray data. Bioinformatics. 2004; 20:917–923. [PubMed: 14751970]

Park T, Yi S, Lee S, Lee SY, Yoo D, Ahn J, Lee Y. Statistical tests for identifying differentially expressed genes in time-course microarray experiments. Bioinformatics. 2003; 19:694–703. [PubMed: 12691981]

Peddada SD, Lobenhofer EK, Li L, Afshari CA, Weinberg CR, Umbach DM. Gene selection and clustering for time-course and dose response microarray experiments using order-restricted inference. Bioinformatics. 2003; 19:834–841. [PubMed: 12724293]

Pitman J. Some probabilistic aspects of set partitions. Am Math Mon. 1997; 104:201–209.

Ruppert, D.; Wand, MP.; Caroll, RJ. Semiparametric regression. New York: Cambridge University Press; 2003.

Schliep A, Schonhuth A, Steinhoff C. Using hidden Markov models to analyze gene expression time course data. Bioinformatics. 2003; 19:i255–i263. [PubMed: 12855468]

Thalamuthu A, Mukhopadhyay I, Zheng X, Tseng GC. Evaluation and comparison of gene clustering methods in microarray analysis. Bioinformatics. 2006; 22:2405–2412. [PubMed: 16882653]

Tibshirani R. Regression shrinkage and selection via the lasso. J R Stat Soc B. 1996; 58:267–288.

Tseng GC, Wong WH. Tight clustering: a resampling-based approach for identifying stable and tight patterns in data. Biometrics. 2005; 61:10–16. [PubMed: 15737073]

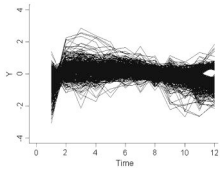

Fig. 1.
Average gene-expressional profiles

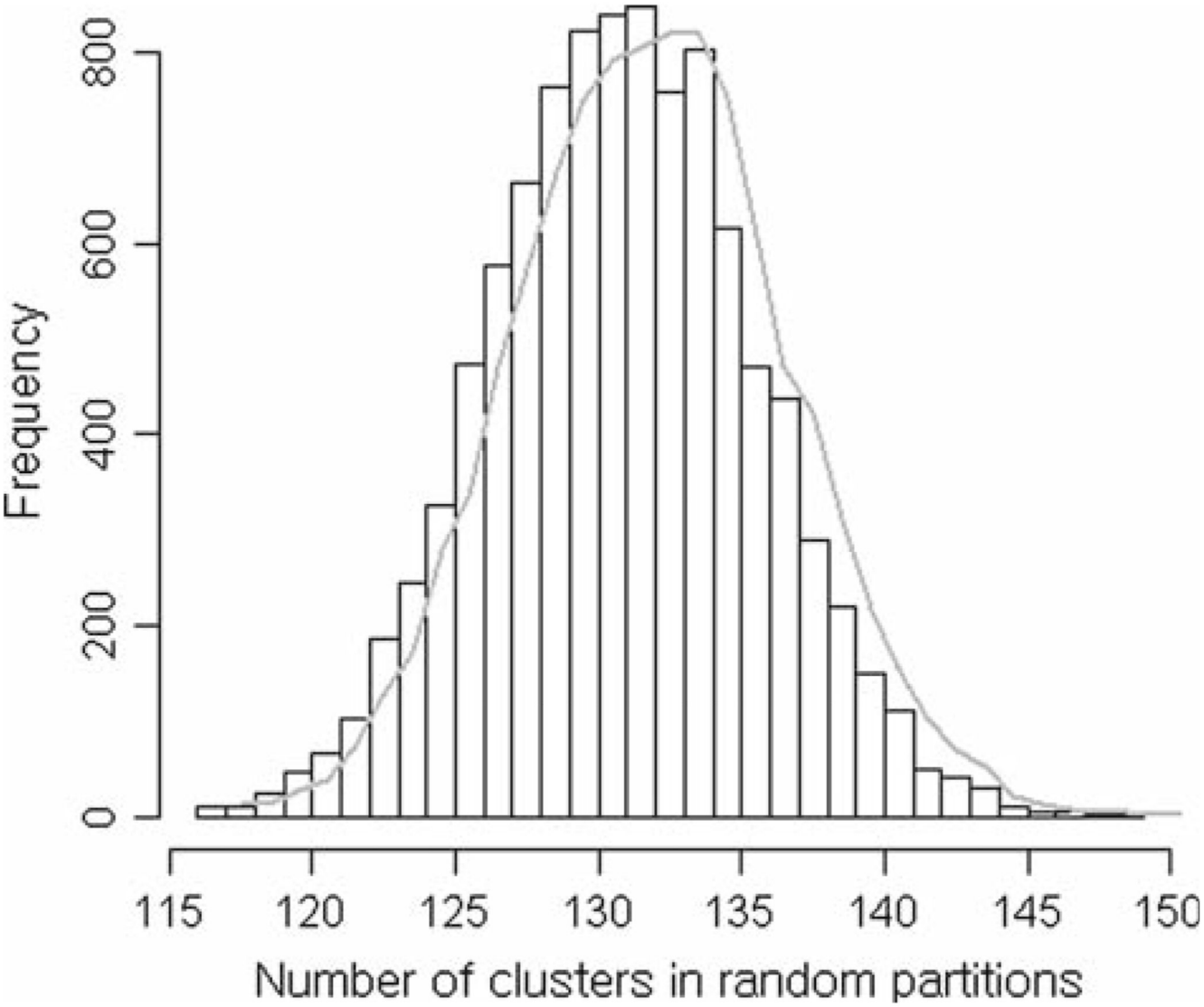

Fig. 2.
Histogram of uniform random partitions. Distribution of *m* in (11) is drawn with a gray line

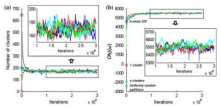

Fig. 3.

Convergence of chains with different initial partitions. **a**, **b** demonstrates the convergence with the number of clusters and the objective function Obj(ω). Simulation history in the last 2×10^6 iterations is magnified in an insert of each figure

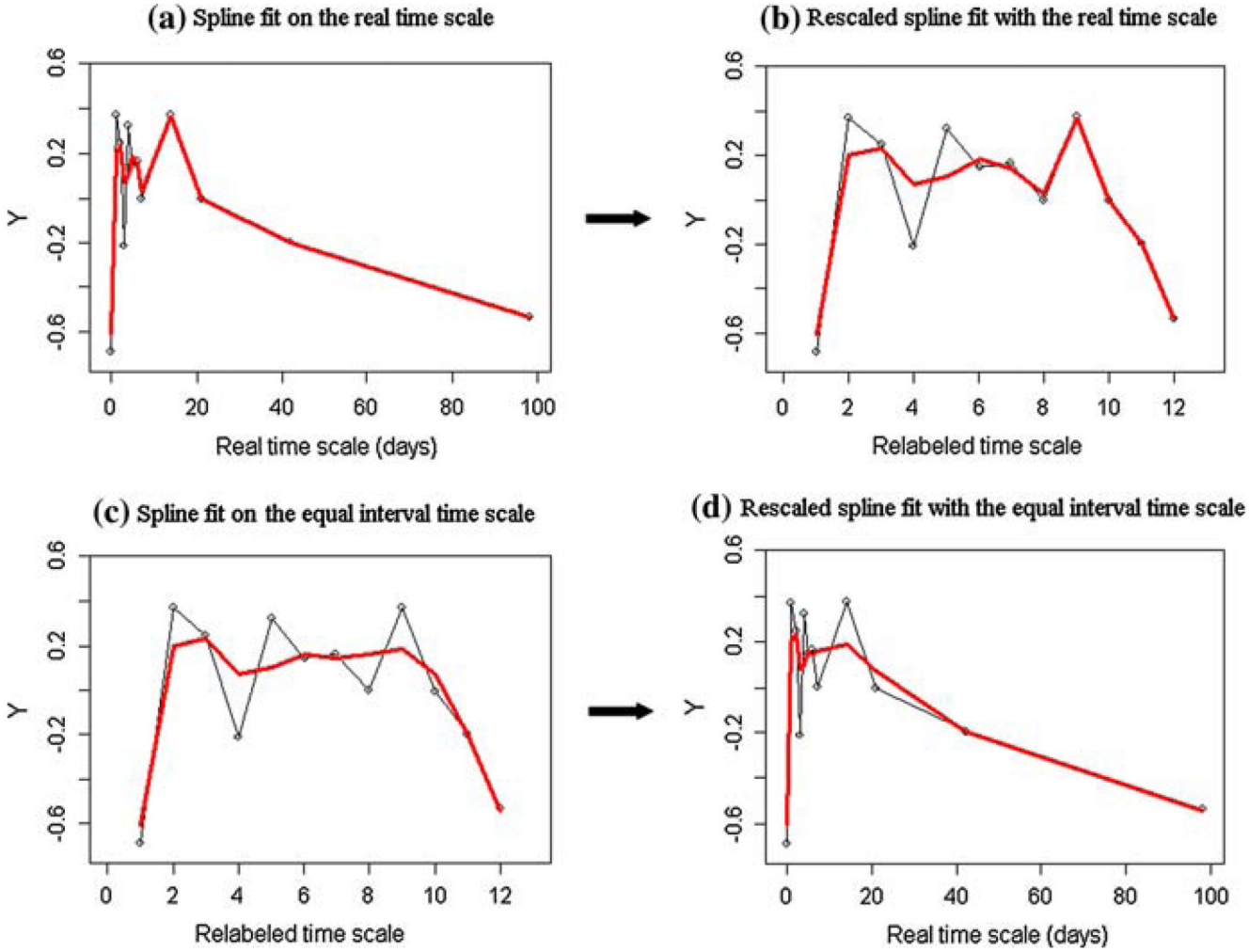

Fig. 4.

Comparisons of spline fits of a randomly selected gene profile when the real time scale or the relabeled equal interval time scale is used: (**a**) gene profile is fitted on the real time scale (days), (**b**) plot **a** is redrawn with the relabeled time scale, (**c**) the profile is fitted on the equal interval scale, and (**d**) plot **c** is redrawn with the relabeled time scale

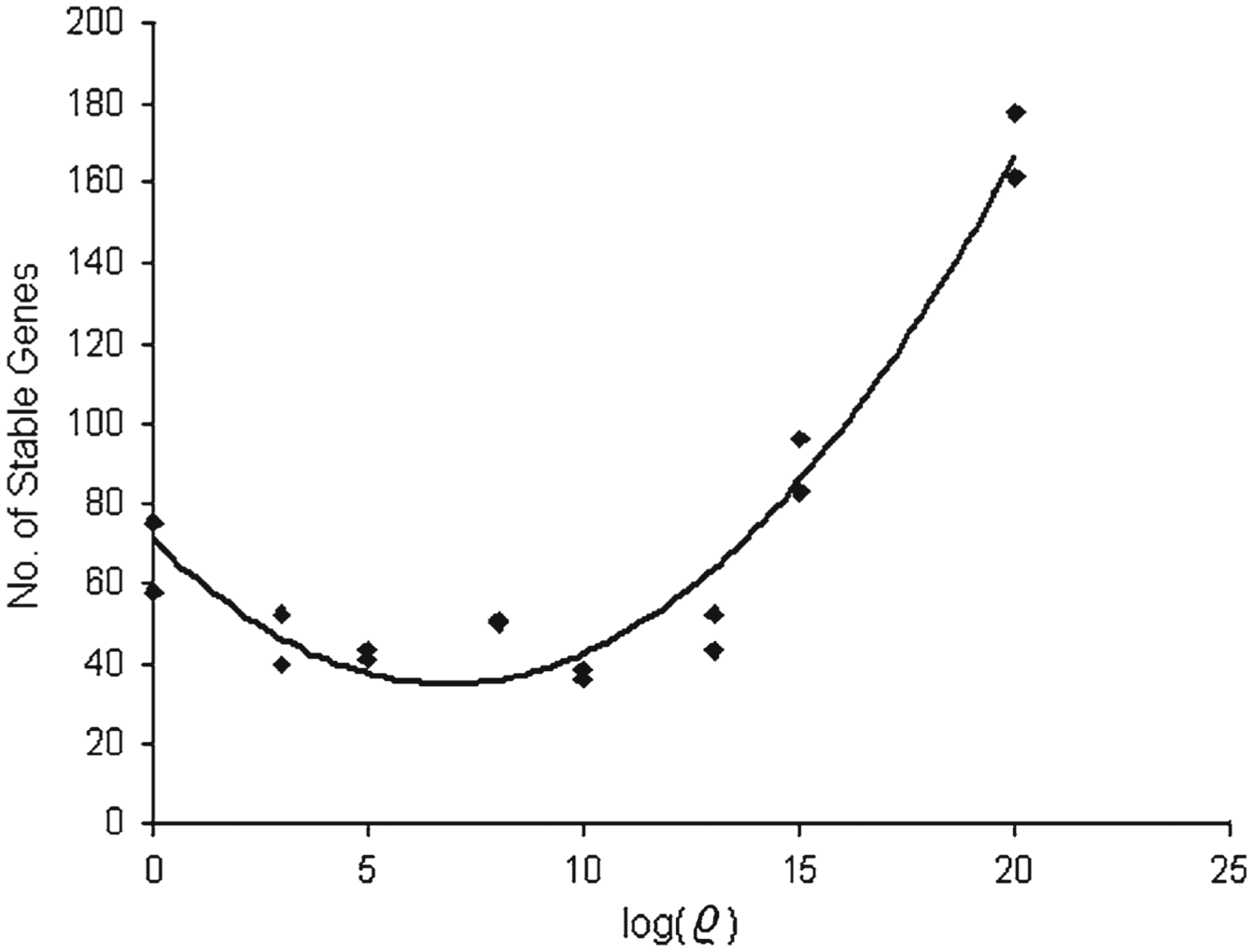

Fig. 5.
The effect of log(ϱ) in the Crowley prior on the number of stable genes. A quadratic regression fit is provided as a *reference line*

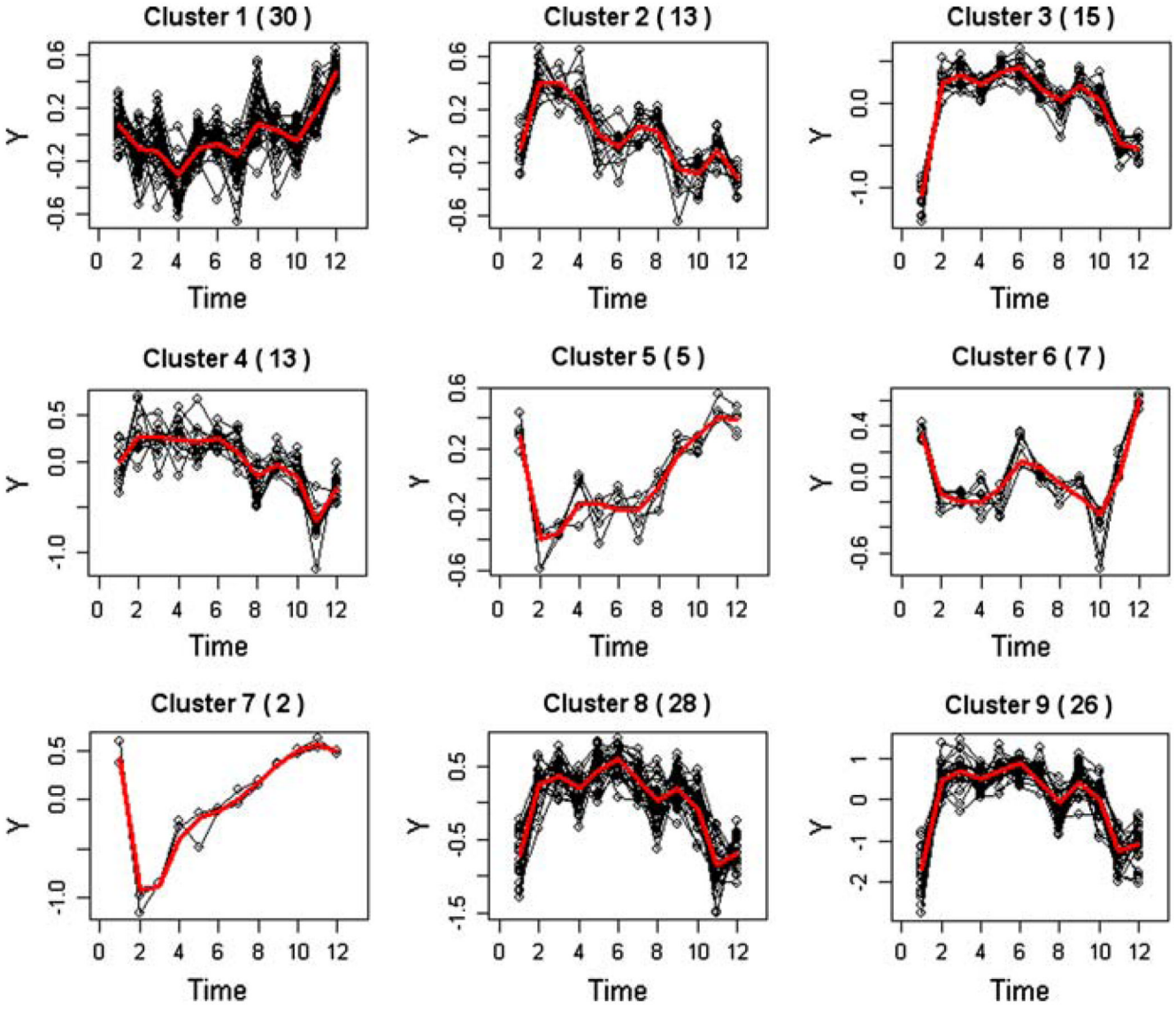

Fig. 6.

Clustered gene profiles of closely related genes. The number of genes in each cluster is described inside parentheses. The light-colored thick lines are from the BLUP of the mixed model (6)

Table 1

Convergence of algorithm with different log(ϱ)'s and stable genes in the last 10^6 iterations

log(ϱ)	**0**		**3**		**5**		**8**	
Run No.	**1st**	**2nd**	**1st**	**2nd**	**1st**	**2nd**	**1st**	**2nd**
Obj($\tilde{\omega}$*)	5,280	5,273	5,338	5,344	5,411	5,404	5,561	5,577
$c(\tilde{\omega}^*)$	27	30	42	44	64	56	135	121
No. of stable genes	58	75	40	52	43	41	50	51
log(ϱ)	10		13		15		20	
Run No.	1st	2nd	1st	2nd	1st	2nd	1st	2nd
Obj($\tilde{\omega}$*)	5,849	5,846	6,940	6,974	8,351	8,342	11,631	11,630
$c(\tilde{\omega}^*)$	224	225	500	510	636	635	645	645
No. of stable genes	36	38	52	43	96	83	162	178

Table 2

Average number of iterations before the optimal partition is found ($Avg(R^*)$)

Simulated data set	$n = 20$	$n = 50$	$n = 100$
1	6,220	4,075,804	$>5 \times 10^7$
2	1,637	7,927,721	$>5 \times 10^7$
3	663	26,765,931	$>5 \times 10^7$
4	4,440	12,796,622	$>5 \times 10^7$
5	934	$>5 \times 10^7$	$>5 \times 10^7$

Table 3

Comparisons of tight and plain Bayes clustering algorithm based on ER: the standard deviation of ER is denoted inside parentheses

σA/σB	0.5/0.5	0.7/0.7	1.0/1.0	0.7/3.0
Tight Clustering	6.0 (0.8)	13.7 (1.3)	26.8 (0.9)	8.8 (0.8)
log(ϱ) = −50	10.3 (0.3)	25.9 (0.9)	40.7 (0.6)	22.4 (0.9)
log(ϱ) = −30	11.1 (0.4)	26.4 (0.9)	40.6 (0.6)	22.7 (0.9)
log(ϱ) = −20	9.8 (0.3)	26.1 (0.9)	40.4 (0.6)	22.0 (0.8)
log(ϱ) = −15	9.8 (0.3)	25.3 (0.9)	40.5 (0.6)	21.5 (0.8)
log(ϱ) = −10	8.8 (0.2)	21.7 (0.8)	39.0 (0.6)	17.4 (0.7)
log(ϱ) = −5	9.8 (0.2)	17.6 (0.5)	29.8 (0.7)	14.9 (0.4)
log(ϱ) = 0	14.6 (0.1)	15.2 (0.1)	16.5 (0.2)	15.1 (0.1)
log(ϱ) = 5	15.3 (0.0)	15.3 (0.0)	15.3 (0.0)	15.3 (0.0)