PReMiU M: An R Package for Profile Regression
Mixture Models using Dirichlet Processes

Silvia Liverani*  
Imperial College London  
and  
MRC Biostatistics Unit  
Cambridge  
* Joint first author  

David I. Hastie*  
Imperial College London  
* Joint first author  

Michail Papathomas  
University of St Andrews  

Sylvia Richardson  
MRC Biostatistics Unit  
Cambridge

Abstract

PReMiU M is a recently developed R package for Bayesian clustering using a Dirichlet process mixture model. This model is an alternative to regression models, non-parametrically linking a response vector to covariate data through cluster membership (Molitor et al. 2010). The package allows Bernoulli, Binomial, Poisson, Normal and categorical response, as well as Normal and discrete covariates. Additionally, predictions may be made for the response, and missing values for the covariates are handled. Several samplers and label switching moves are implemented along with diagnostic tools to assess convergence. A number of R functions for post-processing of the output are also provided. In addition to fitting mixtures, it may additionally be of interest to determine which covariates actively drive the mixture components. This is implemented in the package as variable selection.

Keywords: Profile regression, Clustering, Dirichlet process mixture model.

1. Introduction

Profile regression is an alternative to regression models for when one tries to make inference beyond main effects for datasets with potentially correlated covariates. In particular, profile regression non-parametrically links a response vector to covariate data through cluster membership (Molitor et al. 2010). We have implemented this method in the R (R Core Team 2012) package PReMiU M.

PReMiU M performs Bayesian clustering using a Dirichlet process mixture model and it allows Bernoulli, Binomial, Poisson, Normal and categorical response, as well as Normal and discrete covariates. Moreover, predictions may be made for the response, and missing values for the covariates are handled. Several samplers and label switching moves are implemented along with diagnostic tools to assess convergence. A number of R functions for post-processing of
the output are also provided. In addition to fitting mixtures, it may additionally be of interest
to determine which covariates actively drive the mixture components. This is implemented
in the package as variable selection.

In order to demonstrate the \texttt{PReMiuM} package, it is helpful to present an overview of the
Dirichlet Process. We begin this section by re-familiarising the reader with such a process,
introducing notation that we shall call upon throughout the paper. Formally, let \((\Omega, \mathcal{F}, P)\)
be a probability space comprising a state space \(\Omega\) with associated \(\sigma\)-field \(\mathcal{F}\) and a probability
measure \(P_0\). We say that a probability measure \(P\) follows a Dirichlet process with concentra-
tion parameter \(\alpha\) and base distribution \(P_{\Theta_0}\) parametrised by \(\Theta_0\), written
\(P \sim \text{DP}(\alpha, P_{\Theta_0})\) if
\[
(P(A_1), P(A_2), \ldots, P(A_r)) \sim \text{Dirichlet}(\alpha P_{\Theta_0}(A_1), \alpha P_{\Theta_0}(A_2), \ldots, \alpha P_{\Theta_0}(A_r))
\] (1)
for all \(A_1, A_2, \ldots, A_r \in \mathcal{F}\) such that \(A_i \cap A_j = \emptyset\) for all \(i \neq j\) and \(\bigcup_{j=1}^r A_j = \Omega\).

1.1. The stick-breaking construction

Although Definition (1) is perhaps rather abstract, proof of the existence of such a process
has been determined in a variety of ways, using a number of different formulations (Ferguson,
1973 and Blackwell and MacQueen, 1973). In this paper we focus on Dirichlet process mixture
models (DPMM), based upon the following simplified constructive definition of the Dirichlet
process, due to Sethuraman (1994). If
\[
P = \sum_{c=1}^{\infty} \psi_c \delta_{\Theta_c},
\]
\[
\Theta_c \sim P_{\Theta_0} \text{ for } c \in \mathbb{Z}^+,
\]
\[
\psi_c = V_c \prod_{l<c} (1 - V_l) \text{ for } c \in \mathbb{Z}^+ \setminus \{1\},
\]
\[
\psi_1 = V_1, \text{ and }
\]
\[
V_c \sim \text{Beta}(1, \alpha) \text{ for } c \in \mathbb{Z}^+,
\]
where \(\delta_x\) denotes the Dirac delta function concentrated at \(x\), then \(P \sim \text{DP}(\alpha, P_{\Theta_0})\). This
formulation for \(V\) and \(\psi\) is known as a \textit{stick-breaking} distribution. Importantly, the distri-
bution \(P\) is discrete, because draws \(\Theta_1, \Theta_2, \ldots\) from \(P\) can only take the values in the set
\(\{\Theta_c : c \in \mathbb{Z}^+\}\).

As noted by many authors (for example Ishwaran and James, 2001 and Kalli \textit{et al.}, 2011) it is
possible to extend the above formulation to more general stick-breaking formulations, for
example allowing \(V_c \sim \text{Beta}(a_c, b_c)\) independently, resulting in a generalised Dirichlet process,
such as the two parameter Poisson-Dirichlet process (Pitman and Yor 1997). Another example
is provided by Chung and Dunson (2009) and Rodriguez and Dunson (2011) where the Beta
distributions are replaced by a Probit model, which may be allowed to depend on observed
data, for the purpose of capturing dependence such as a spatial structure. The methods and
results that we propose within this paper hold for such generalised processes, but at present
the package is only coded to implement the Dirichlet process in equation (2).

Typically, because of the complexity of the models based on the stick-breaking construction,
inference is made in a Bayesian framework using Markov chain Monte Carlo (MCMC) meth-
ods. Until recently, a perceived difficulty in making inference about this model was the infinite
number of parameters within the stick breaking construction. Historically, this obstacle has resulted in the use of algorithms that either explore marginal spaces where some parameters are integrated out or use truncated approximations to the full Dirichlet process mixture model, see for example Neal (2000) and Ishwaran and James (2001).

More recently, two alternative innovative approaches to sampling the full DPMM have been proposed. The first, introduced by Walker (2007), uses a novel slice sampling approach, resulting in full conditionals that may be explored by the use of a Gibbs sampler. The difficulty of the proposed approach is the introduction of constraints that complicate the updates of the mixture component weights, leading to potential mixing issues. To overcome this, Kalli et al. (2011) generalise this sampler, adding further auxiliary variables, and report good convergence results, although the authors note that the algorithm is sensitive to these additional parameters. The second distinct MCMC sampling approach was proposed in parallel by Papaspiliopoulos and Roberts (2008). The proposed sampler again uses a Gibbs sampling approach, but is based upon an idea termed retrospective sampling, allowing a dynamic approach to the determination of the number of components (and their parameters) that adapts as the sampler progresses. The cost of this approach is an ingenious but complex Metropolis-within-Gibbs step, to determine cluster membership.

Despite the apparent differences between the two strategies, Papaspiliopoulos (2008) noted that the two algorithms can be effectively combined to yield an algorithm that improves either of the originals. The resulting sampler was implemented and presented by Yau et al. (2011), and a similar version was presented by Dunson (2009) for DPMM. The current sampler presented in this paper is our interpretation of these ideas, implemented as an R package. This package, called PReMiuM, is based upon efficient underlying C++ code for general DPMM sampling and it is available on CRAN.

In Section 2 we describe formally the Dirichlet process mixture model implemented in PReMiuM and the blocked MCMC sampler used. In Section 3 we discuss profile regression and its link with the response and covariate models included in the package. In Section 4 we give an overview of the postprocessing tools available in PReMiuM to learn from the rich output produced by our Bayesian model and in Section 5 we discuss diagnostic tools that we propose to investigate the convergence of the MCMC. Finally, in Section 6 we give a brief overview of the structure of the code and show examples of its use as well as an indication of run times in Section 7.

2. Sampling the Dirichlet process mixture model

2.1. Definition and properties

Perhaps the most common application of the Dirichlet process is in clustering data, where it can be used as the prior distribution for the parameters of an infinite mixture model. Consider again the stick breaking construction in Equation 2. For the Dirichlet process mixture model (DPMM), the (possibly multivariate) observed data $D = (D_1, D_2, \ldots, D_n)$ follow an infinite mixture distribution, where component $c$ of the mixture is a parametric density of the form $f_c(\cdot) = f(\cdot | \Theta_c, \Lambda)$ parametrised by some component specific parameter $\Theta_c$ and some global parameter $\Lambda$. Defining (latent) parameters $\tilde{\Theta}_1, \tilde{\Theta}_2, \ldots, \tilde{\Theta}_n$ as draws from a probability distribution $P$ following a Dirichlet process $DP(\alpha, P_{\Theta_0})$ and again denoting the
dirac delta function by $\delta$, this system can be written,
\begin{align*}
D_i | \tilde{\Theta}_i, \Lambda & \sim f(D_i | \tilde{\Theta}_i, \Lambda) \quad \text{for } i = 1, 2, \ldots, n, \\
\tilde{\Theta}_i & \sim \sum_{c=1}^{\infty} \psi_c \delta_{\Theta_c} \quad \text{for } i = 1, 2, \ldots, n, \\
\Theta_c & \sim P_{\Theta_0} \quad \text{for } c \in \mathbb{Z}^+, \\
\psi_c & = V_c \prod_{l < c} (1 - V_l) \quad \text{for } c \in \mathbb{Z}^+ \setminus \{1\}, \\
\psi_1 & = V_1, \quad \text{and} \\
V_c & \sim \text{Beta}(1, \alpha) \quad \text{for } c \in \mathbb{Z}^+.
\end{align*}

When making inference using mixture models (either finite or infinite) it is common practice to introduce a vector of latent allocation variables $Z$. Such variables enable us to explicitly characterise the clustering and additionally facilitate the design of MCMC samplers. Adopting this approach and writing $\psi = (\psi_1, \psi_2, \ldots)$ and $\Theta = (\Theta_1, \Theta_2, \ldots)$, we re-write Equation 3 as
\begin{align*}
D_i | Z, \Theta, \Lambda & \sim f(D_i | \Theta_{Z_i}, \Lambda) \quad \text{for } i = 1, 2, \ldots, n, \\
\Theta_c & \sim P_{\Theta_0} \quad \text{for } c \in \mathbb{Z}^+, \\
P(Z_i = c | \psi) & = \psi_c \quad \text{for } c \in \mathbb{Z}^+, \quad i = 1, 2, \ldots, n, \\
\psi_c & = V_c \prod_{l < c} (1 - V_l) \quad \text{for } c \in \mathbb{Z}^+ \setminus \{1\}, \\
\psi_1 & = V_1, \quad \text{and} \\
V_c & \sim \text{Beta}(1, \alpha) \quad \text{for } c \in \mathbb{Z}^+.
\end{align*}

The likelihood of $D_i$ associated with the DPMM is simply the first line of Equation 4. Integrating out the latent variable $Z_i$ we obtain the more recognisable mixture likelihood
\begin{align*}
p(D_i | \psi, \Theta, \Lambda) & = \sum_{c=1}^{\infty} p(D_i | Z_i = c, \Theta, \Lambda) p(Z_i = c | \psi) \\
& = \sum_{c=1}^{\infty} \psi_c f(D_i | \Theta_c, \Lambda).
\end{align*}

The remainder of Equation 4 provides the prior specification for the DPMM, allowing us to write the joint posterior distribution as
\begin{align*}
p(Z, \Theta, \Lambda, V, \alpha | D) & \propto p(D | Z, \Theta, \Lambda) p(Z, V, \alpha, \Theta, \Lambda | \Theta_0) \\
& \propto \prod_{i=1}^{n} \{f(D_i | \Theta_{Z_i}, \Lambda) p(Z_i | V)\} \prod_{c=1}^{\infty} \{p(V_c | \alpha) p(\Theta_c | \Theta_0)\} p(\Lambda) p(\alpha) \\
& \propto \prod_{i=1}^{n} \left\{ f(D_i | \Theta_{Z_i}, \Lambda) \left[ V_{Z_i} \prod_{l < Z_i} (1 - V_l) \right] \right\} \\
& \quad \times \prod_{c=1}^{\infty} \{ (1 - V_c)^{\alpha-1} p(\Theta_c | \Theta_0) \} p(\Lambda) p(\alpha).
\end{align*}
Of course, additional layers of hierarchy could easily be introduced, for example through hyper-priors for $\Theta_0$.

2.2. The MCMC sampler

A common approach to MCMC sampling from the DPMM is to integrate out $V$ and use a Gibbs sampler on the resulting space. Such samplers are commonly referred to as Pólya urn samplers, since they are motivated by the Pólya urn representation of a Dirichlet process introduced by Blackwell and MacQueen (1973). Ishwaran and James (2001) provide a review of this approach and demonstrate that conditionals of the type $p(Z_i|Z_{-i})$ can be derived, where $Z_{-i} = (Z_1, \ldots, Z_{i-1}, Z_{i+1}, Z_n)$. Many other authors have focused on developing alternative samplers of this nature, including Neal (2000) and Green (2010). However, as noted by many authors, samplers of this nature, where the allocation of a single observation is conditional on the allocations of all other observations, can often suffer from poor mixing. This motivates the need for an alternative class of samplers that sample from the full model.

In the full model, the posterior conditionals for the allocation variables $Z$ depend upon an infinite number of variables $V$ and $\Theta$. One way to bypass this complication is to truncate the definition in Equation 4 to mixtures with $C$ components (of which potentially only a subset will be non-empty). Ishwaran and James (2001) demonstrate that under this model the conditional distributions are standard distributions which can be easily sampled from. This is one of the three samplers implemented in our R package, and we refer to it as truncated. Although this approach resolves the challenges of sampling from the full model, if $C$ is not chosen to be sufficiently large, then the posterior may be quite different on the truncated model space compared to the full model space. The authors provide some guidance for choice of $C$, but more recent work by Walker (2007) and Papaspiliopoulos and Roberts (2008) demonstrate techniques that alleviate the need for such a truncation, whilst retaining many of the sampling properties of the full conditionals.

The first step is to borrow the idea of Walker (2007) and introduce auxiliary variables $U = (U_1, U_2, \ldots, U_n)$ such that the joint posterior can be re-written as

$$p(U, Z, \Theta, \Lambda, V, \alpha|D) \propto \prod_{i=1}^{n} \left\{ f(D_i|\Theta_{Z_i}, \Lambda) \mathbf{1}_{\{U_i<\psi_{Z_i}\}} \right\} \times \prod_{c=1}^{\infty} \left\{ (1 - V_c)^{\alpha-1} p(\Theta_c|\Theta_0) \right\} p(\Lambda)p(\alpha)$$

$$\propto \prod_{i=1}^{n} \left\{ f(D_i|\Theta_{Z_i}, \Lambda) \mathbf{1}_{\{U_i<V_{Z_i}, \prod_{c<V_{Z_i}} (1-\psi_{Z_i})\}} \right\} \times \prod_{c=1}^{\infty} \left\{ (1 - V_c)^{\alpha-1} p(\Theta_c|\Theta_0) \right\} p(\Lambda)p(\alpha).$$

Here, $\mathbf{1}_A$, is the function that takes the value 1 over the set $A$ and 0 elsewhere. By combining this auxiliary variable approach with the notion of retrospective sampling (i.e. adopting a just-in-time approach to sampling empty mixture component parameters, as introduced in Papaspiliopoulos and Roberts, 2008), it is possible to construct an efficient Gibbs sampler for sampling from the joint posterior. Integrating $U$ out of Equation 7 with respect to the Lebesgue measure yields the DPMM posterior distribution given in Equation 5 meaning that
marginalising samples of the joint distribution over $U$ results in samples from the desired distribution.

Kalli et al. (2011) extend the idea of Walker (2007) to a general class of slice samplers by writing

$$ p(U, Z, \Theta, \Lambda, V, \alpha | D) \propto \prod_{i=1}^{n} \left\{ f(D_i | \Theta Z_i, \Lambda) 1_{U_i < \xi Z_i} \xi^{-1}_Z \right\} \prod_{c=1}^{\infty} \left\{ (1 - V_c)^{\alpha - 1} p(\Theta_c | \Theta_0) \right\} p(\Lambda) p(\alpha) $$

(8)

where $\xi_1, \xi_2, \ldots$ is any positive sequence.

When $\xi_i = \psi_i$, this corresponds to the efficient Gibbs sampler proposed by Papaspiliopoulos (2008) in the context of DPMM using parameter blocking. This is the second of the three samplers implemented in our R package, and we refer to it as slice dependent, in accordance with Kalli et al. (2011). Most importantly, this design permits the introduction of label switching moves, without which it is very difficult to obtain sufficient mixing. We discuss this in detail in Section 5.

For the last of the three samplers implemented in our R package, that we refer to as slice independent in accordance with Kalli et al. (2011), we set $\xi = (\kappa - 1)j^{\kappa}$ with $\kappa = 0.8$ as proposed by Kalli et al. (2011).

We continue by defining some new notation which is required to present our slice samplers. First, given the allocation variables $Z$, define

$$ Z^* = \max_{1 \leq i \leq n} Z_i. $$

Similarly, given the auxiliary variables $U$ and the vector $V$, define

$$ U^* = \min_{1 \leq i \leq n} U_i. $$

and

$$ C^* = \min \left\{ c \in \mathbb{Z}^+ : \sum_{l=1}^{c} \psi_l > 1 - U^* \right\} = \min \left\{ c \in \mathbb{Z}^+ : \sum_{l=1}^{c} \left[ V_l \prod_{r<l} (1 - V_r) \right] > 1 - U^* \right\}. $$

(9)

It is important to emphasise that these values potentially change at each sweep of the sampler, as the underlying variables change, although for simplicity of exposition we have omitted explicitly labelling the parameters with the sweep. The purpose of the variable $C^*$ is to provide an upper limit on which mixture components need updating at each sweep. Specifically, although there are infinitely many component parameters in the model, since $P(Z_i = c | U_i > \psi_c) = 0$, we need only concentrate our updating efforts on those components $c$ for which $\psi_c > U_i$ for some $i = 1, 2, \ldots, n$. By defining $C^*$ as in equation 9 it can be shown (see Appendix A.1) that $\psi_c < U_i$ for all $c > C^*$ and all $i = 1, 2, \ldots, n$. Assuming that the Markov chain is initialised accordingly and is updated using the correct conditionals, it is possible to show $Z^* \leq C^* < \infty$ almost surely (details provided in Appendix A.1).
With these definitions in place we make use of the following sets and vectors (which again will change at each sweep)

\[ A = \{ c \in \mathbb{Z}^+ : c \leq Z^* \}, \quad P = \{ c \in \mathbb{Z}^+ : Z^* < c \leq C^* \}, \quad I = \{ c \in \mathbb{Z}^+ : c > C^* \} \]

\[ V^A = (V_1^A, V_2^A, \ldots, V_{Z^*}^A), \quad \Theta^A = (\Theta_1, \Theta_2, \ldots, \Theta_{Z^*}) \]

\[ V^P = (V_{Z^*+1}^P, V_{Z^*+2}^P, \ldots, V_{C^*}^P), \quad \Theta^P = (\Theta_{Z^*+1}, \Theta_{Z^*+2}, \ldots, \Theta_{C^*}) \]

\[ V^I = (V_{C^*+1}, V_{C^*+2}, \ldots), \quad \Theta^I = (\Theta_{C^*+1}, \Theta_{C^*+2}, \ldots). \]

Here the A, P and I are disjoint sets (updated at every sweep of the MCMC algorithm) that partition \( \mathbb{Z}^+ \), with names chosen to denote Active, Potential and Inactive components respectively. It is possible that \( P = \emptyset \). By definition, all observations are allocated to a mixture component labelled by one of the indices in A. Components labelled with an index in P or I are necessarily empty (i.e. have no observations allocated to them), the difference being that at the next update of the allocation variables, components with labels in P may potentially become non-empty.

The blocked infinite DPMM algorithm can now be defined using the following blocked Gibbs updates to sample from the relevant conditionals.

**DPMM algorithm**

Suppose we are at sweep \( t \) of the sampler. Update as follows:

**Step A.** Compute \( Z^* \) and the set \( A \).

**Step B.** Sample \((V^A_{t+1}, \Theta^A_{t+1}, \tilde{Z}, U_{t+1}) \sim p(V^A, \Theta^A, Z, U|V^P_t, V^I_t, \Theta^P_t, \Theta^I_t, \alpha_t, \Lambda_t, \Theta_0, D)\).

**B.1** \( \tilde{V}^A \sim p(V^A|Z_t, \alpha_t) \)

**B.2** \( \Theta^A \sim p(\Theta^A|Z_t, \Lambda_t, \Theta_0, D) \)

**B.3** \((V^A_{t+1}, \Theta^A_{t+1}, \tilde{Z}) \sim p(V^A, \Theta^A, Z|V^P_t, \Theta^P_t, \Theta^I_t, \alpha_t, \Lambda_t, \Theta_0, D) \)

**B.4** \( U_{t+1} \sim p(U|V^A_{t+1}, \tilde{Z}) \)

**Step C.** Compute \( U^* \). Recompute \( Z^* \) and the set \( A \).

**Step D.** Sample \((\alpha_{t+1}, V^P_{t+1}, V^I_{t+1}) \sim p(\alpha, V^P, V^I|V^A_{t+1}, \Theta^A_{t+1}, \Theta^I_{t+1}, U_{t+1}, \tilde{Z}, \Lambda_t, \Theta_0, D)\), computing \( C^* \) and the set \( P \) in the process.

**D.1** \( \alpha_{t+1} \sim p(\alpha|V^A_{t+1}) \)

**D.2** \( V^P_{t+1} \sim p(V^P|\alpha_{t+1}) \)

**Step E.** Sample \((\Theta^P_{t+1}, \Theta^I_{t+1}) \sim p(\Theta^P, \Theta^I|V^A_{t+1}, V^P_{t+1}, V^I_{t+1}, \Theta^A_{t+1}, \tilde{Z}, \alpha_{t+1}, \Lambda_t, \Theta_0, D)\).

**E.1** \( \Theta^P_{t+1} \sim p(\Theta^P|\Theta_0) \)

**Step F.** Sample \( \Lambda_{t+1} \sim p(\Lambda|V^A_{t+1}, V^P_{t+1}, V^I_{t+1}, \Theta^A_{t+1}, \Theta^P_{t+1}, \Theta^I_{t+1}, \tilde{Z}, \alpha_{t+1}, \Theta_0, D) \).

**F.1** \( \Lambda \sim p(\Lambda|\Theta^A_{t+1}, \tilde{Z}, D) \)

**Step G.** Sample \( Z_{t+1} \sim p(Z|V^A_{t+1}, V^P_{t+1}, V^I_{t+1}, \Theta^A_{t+1}, \Theta^P_{t+1}, \Theta^I_{t+1}, U_{t+1}, \alpha_{t+1}, \Lambda_{t+1}, \Theta_0, D) \).
While this algorithm is somewhat generic, the blocking strategy is clearly highlighted. Further details explaining each of the steps are provided in Appendix A.2. The key idea is that by doing joint updates, we can marginalise out an infinite number of variables when necessary, to ensure that we are always sampling from conditional distributions that depend only upon a finite number of parameters. In particular, after marginalisation, the parameters corresponding to the inactive set \( I \) do not contribute to the conditional distributions of the other parameters, so we do not actually need to sample their values. Since these parameters have no contribution to the likelihood, if values are subsequently required they can simply be sampled from the prior retrospectively as necessary. Although the sampler is written as a blocked Gibbs sampler, where it is not possible to sample directly from full conditionals (for example in the update of \( \Theta \), depending upon the choices of \( f \) and \( P_{\Theta_0} \)) Metropolis-within-Gibbs steps are applied. Depending on the application, the Gibbs updates specified above may comprise several different Gibbs or Metropolis-within-Gibbs steps (for example updating \( \Theta \) and \( \Lambda \). Typically, where Metropolis-Hastings updates are required we advocate adopting an adaptive Metropolis-Hastings approach: see Andrieu and Thoms (2008) for a review.

3. Example Models

The general sampler of the previous section is applicable for many specific models, depending on the choices of \( f \) and \( P_{\Theta_0} \). In this section we provide further details for some of the models that are implemented within our software. We detail the prior choices that are made within our implementation, but, of course, alternative priors could be chosen.

3.1. Gaussian mixtures

Perhaps the most common model to be implemented under the DPMM framework is the Gaussian mixture model, where \( D = X \) for some covariate data \( X \), and \( X \) assumes a mixture of Gaussian distributions. Under this setting for each cluster \( c \), the cluster specific parameters are given by \( \Theta_c = (\mu_c, \Sigma_c) \), where \( \mu_c \) is a mean vector and \( \Sigma_c \) is a covariance matrix. There are no additional global parameters \( \Lambda \). Under this setting

\[
p(X_i|Z_i, \Theta_{Z_i}, \Lambda) = f(X_i|\mu_{Z_i}, \Sigma_{Z_i}) = (2\pi)^{-\frac{d}{2}}|\Sigma_{Z_i}|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(X_i - \mu_{Z_i})^T \Sigma_{Z_i}^{-1}(X_i - \mu_{Z_i})\right\}.
\]

By choosing \( \mu_c \sim \text{Normal}(\mu_0, \Sigma_0) \) and \( \Sigma_c \sim \text{InvWishart}(R_0, \kappa_0) \) (for each \( c \)) for our prior model \( P_{\Theta_0} \) we have a conjugate model, permitting Gibbs updates for the parameters \( \mu^A \) and \( \Sigma^A \) associated the active components, and also those \( (\mu^P, \Sigma^P) \) associated with the potential components. The choice of values for the hyperparameters \( \Theta_0 = (\mu_0, \Theta_0, R_0, \kappa_0) \) is discussed further in section 6.

3.2. Discrete mixtures

Clearly the DPMM model applies to mixtures other than simply the Gaussian case. Consider for example the case where for each individual \( i \), \( D_i = X_i \) is a vector of \( J \) independent discrete categorical random variables, where the number of categories for covariate \( j = 1, 2, \ldots, J \) is
Then we can write \( \Theta_c = \Phi_c = (\phi_{c,1,1}, \phi_{c,1,2}, \ldots, \phi_{c,1,K_1}, \phi_{c,2,1}, \ldots, \phi_{c,2,K_2}, \phi_{c,J_1,1}, \ldots, \phi_{c,J_1,K_J}) \) and

\[
p(D_i|Z_i, \Theta_{Z_i}, \Lambda) = f(D_i|\Phi_{Z_i}) = \prod_{j=1}^{J} \phi_{Z_i,j,X_{i,j}}.
\]

Again, there are no global parameters \( \Lambda \).

Letting \( \Theta_{0} = a = (a_1, a_2, \ldots, a_J) \), where for \( j = 1, 2, \ldots, J \), \( a_j = (a_{j,1}, a_{j,2}, \ldots, a_{j,K_j}) \) and adopting conjugate Dirichlet priors \( \Phi_{c,j} = (\phi_{c,j,1}, \phi_{c,j,2}, \ldots, \phi_{c,j,K_j}) \sim \text{Dirichlet}(a_j) \), each \( \Phi_c \) can be updated directly using Gibbs updates. For full details of the posterior conditional distribution see Molitor et al. (2010).

### 3.3. Mixed mixtures

An alternative model is given by a mixture of some continuous and discrete random variables. Following the notation used above for Gaussian and discrete mixtures, for \( J_1 \) continuous random variables and \( J_2 \) discrete random variables,

\[
p(D_i|Z_i, \Theta_{Z_i}, \Lambda) = p(D_{i1}|\mu_{Z_i}, \Sigma_{Z_i})p(D_{i2}|\Phi_{Z_i}),
\]

where \( D_{i1} \) is the subset of the continuous random variables in \( D_i \) and \( D_{i2} \) is the subset of the categorical random variables in \( D_i \). Note that we are assuming independence between continuous and categorical data conditional on the cluster allocations.

### 3.4. Profile regression

Recently, interest has grown in using DPMM as an alternative to regression models, non-parametrically linking a response vector \( Y \) to covariate data \( X \) through cluster membership. This idea has been pioneered by several authors including Dunson et al. (2008), Bigelow and Dunson (2009), Molitor et al. (2010), Papathomas et al. (2011), and Molitor et al. (2011). Our presentation is most similar to the latter three of these articles which refer to this idea as “profile regression”.

For the case of either Gaussian or Discrete mixtures, as described above, our implementation permits the joint modelling of a response vector \( Y \) and covariate data \( X \) through cluster membership. Formally, the data \( D = (Y, X, W) \) is now extended to contain response data \( Y_i \) and covariate data \( X_i \) for each individual \( i \), where the contribution of the covariate data to the response may be cluster dependent. There is also the possibility to include additional fixed effects \( W_i \) for each individual, which are constrained to only have a global (i.e. non-cluster specific) effect on the response \( Y_i \).

The data \( D_i \) is then jointly modelled as the product of a response model and a covariate model, to give the following likelihood:

\[
p(D_i|Z_i, \Theta, \Lambda) = f_Y(Y_i|\Theta_{Z_i}, \Lambda, W_i)f_X(X_i|\Theta_{Z_i}, \Lambda).
\]

The covariate likelihood \( f_X \) is of either of the forms presented in Sections 3.1 or 3.2. The likelihood \( f_Y \) depends upon the choice of response model.

**Binary response**

Adopting a binary response model, each parameter vector \( \Theta_c \) is extended to include an additional parameter \( \theta_c \). We also introduce the global parameter vector \( \Lambda = \beta \), of the same
length $L$ as the fixed effects vector $W_i$, to capture the contribution of these effects. Then, 

$$f_Y(Y_i|\Theta_{Z_i},\Lambda,W_i) = p(Y_i = 1|\theta_{Z_i},\beta,W_i)$$

is modelled as

$$\logit\{p(Y_i = 1|\theta_{Z_i},\beta,W_i)\} := \lambda_i = \theta_{Z_i} + \beta^T W_i.$$ 

For each cluster $c$, we adopt a $t$ location-scale distribution for $\theta_c$, with hyperparameters $\mu_\theta$ and $\sigma_\theta$ with 7 degrees of freedom, as discussed by Molitor et al. (2010). Similarly, for each fixed effect $l$, we adopt the same prior for $\beta_l$, but with hyperparameters $\mu_\beta$ and $\sigma_\beta$.

The components of $\Theta_c$ corresponding to the covariate model for $X$ retain the possibility of being updated according to Gibbs samples. However, since conjugacy is not achieved with our prior choice, updating $\theta_c$ for each cluster (and $\beta_l$ for each fixed effect $l$) requires a Metropolis-within-Gibbs sample. In our implementation we propose the use of adaptive random-walk-Metropolis moves.

**Categorical response**

The categorical response model that we use is a simple extension of the binary response model of the previous section. In particular, each parameter vector $\Theta_c$ additionally contains an extra parameter vector $\theta_c = (\theta_{c,1}, \theta_{c,2}, \ldots, \theta_{c,R-1})$ of length $R - 1$, where $R$ is the number of possible categories represented in the response data $Y$. Treatment of fixed effects is also extended, so that for each response category $r = 1, 2, \ldots, R - 1$, there is a vector $\beta_r = (\beta_{r,1}, \beta_{r,2}, \ldots, \beta_{r,L})$, where $\beta_{r,l}$ is the coefficient for each fixed effect $l$ ($l = 1, 2, \ldots, L$). This gives $f_Y(Y_i|\Theta_{Z_i},\Lambda,W_i) = p(Y_i = r|\theta_{Z_i,r},\beta,W_i)$ as

$$\logit\{p(Y_i = r|\theta_{Z_i,r},\beta,W_i)\} = \theta_{Z_i,r} + \beta_r^T W_i,$$

for $r = 1, 2, \ldots, R - 1$ and $p(Y_i = 0|\theta_{Z_i,r},\beta,W_i) = 1 - \sum_{r=1}^{R-1} p(Y_i = r|\theta_{Z_i,r},\beta,W_i)$.

In our sampler we use the same priors for each $\theta_{c,r}$ as for $\theta_c$ and $\beta_{r,l}$ as for $\beta_l$ in the binary case, with the resulting observation about Metropolis-within-Gibbs updates remaining true.

**Binomial response**

By providing a number of trials $T_i$ associated with each individual $i$ (in this model an “individual” might correspond to an area or “experiment”) we can extend the binary response model to a Binomial response model. In particular,

$$f_Y(Y_i|\Theta_{Z_i},\Lambda,W_i) = p(Y_i|\theta_{Z_i},\beta,W_i) = \left(\frac{T_i}{Y_i}\right)^{Y_i} \left(1 - \frac{T_i}{Y_i}\right)^{T_i - Y_i},$$

where

$$\logit\{p_i\} := \lambda_i = \theta_{Z_i} + \beta^T W_i.$$

Priors used are identical to the binary case. Molitor et al. (2011) provide an example where this model is employed.

**Poisson response**

For count-type response data, an alternative to the Binomial model is the Poisson model. Under this model, each individual $i$ is associated with an expected offset $E_i$, and the response is then modelled as

$$f_Y(Y_i|\Theta_{Z_i},\Lambda,W_i) = p(Y_i|\theta_{Z_i},\beta,W_i) = \frac{\mu_i^{Y_i}}{Y_i!} \exp\{-\mu_i\},$$

where $\mu_i$ is associated with an expected offset $E_i$. Molitor et al. (2011) provide an example where this model is employed.
where
\[
\mu_i = E_i \exp\{\lambda_i\}, \text{ for } \lambda_i = \theta Z_i + \beta^T W_i.
\]

Prior models for \(\theta_c\) and \(\beta\) are as above.

**Extra variation in the response**

For some of the above models, it is possible that we may wish to allow for extra variation in the response. Our sampler is designed to achieve this by alternatively modelling \(\lambda_i\) (as defined in the above response models) by

\[
\lambda_i = \theta Z_i + \beta^T W_i + \varepsilon_i, \text{ where } \varepsilon_i \sim \text{Normal}(0, \sigma^2_\varepsilon).
\]

Prior distributions for \(\theta_c\) and \(\beta\) are unchanged, but in this model \(\Lambda\) contains an additional parameter, \(\sigma^2_\varepsilon\), for which prior specification is required. For simplicity we work in terms of the precision \(\tau_\varepsilon = 1/\sigma^2_\varepsilon\), and adopt a gamma distribution with shape parameter \(s_\tau_\varepsilon\) and rate parameter \(r_\tau_\varepsilon\). This approach permits a simple Gibbs update of this parameter. In order to make inference about this model, it is also necessary to update the latent variables \(\lambda_i\) at every sweep of the MCMC sampler. These parameters are considered an extension of \(\Lambda\) (as they are not directly associated with a specific cluster) and are therefore updated in Step F of the DPMM algorithm. Updates to these parameters are done using adaptive Random-walk-Metropolis steps.

**Gaussian response**

Our sampler is able to handle Normally distributed continuous response data. As for many of the discrete response models, \(\Theta_c\) is extended to contain \(\theta_c\) for each \(c\). As before \(\Lambda\) contains \(\beta\), but also \(\sigma^2_Y\). These parameters allow us to write the response model as:

\[
f_Y(Y_i|\Theta Z_i, \Lambda, W_i) = p(Y_i|\theta Z_i, \beta, \sigma^2_Y, W_i) = \frac{1}{\sqrt{2\pi\sigma^2_Y}} \exp\left\{-\frac{1}{2\sigma^2_Y} (Y_i - \lambda_i)^2\right\},
\]

where \(\lambda_i = \theta Z_i + \beta^T W_i\).

We impose the same prior settings as for the discrete response models, with the additional prior on \(\tau_Y = 1/\sigma^2_Y\) being Gamma\((s_\tau_Y, r_\tau_Y)\), where \(s_\tau_Y\) and \(r_\tau_Y\) are the shape and rate hyper parameters that extend \(\Theta_0\). Adopting this conjugate prior, updates for \(\tau_Y\) are simple Gibbs updates.

### 3.5. Variable selection

In addition to fitting mixtures, potentially linking covariates and responses, it may additionally be of interest to determine which covariates actively drive the mixture components, and which share characteristics common to all components. This can be formulated as a question of variable selection. Below we present details of how this idea can be modelled, first in the case of discrete covariates, as considered by Papathomas *et al.* (2012), and then in the context of Gaussian covariates, a formulation which, as far as we are aware, has not been reported elsewhere. Variable selection for DPMM has also been considered by Chung and Dunson (2009).
Discrete covariates

Following the approach taken by Papathomas et al. (2012) our sampler implements two types of variable selection. We outline these approaches briefly in this section but for full details the reader is referred to this paper.

The first approach is a type of “soft” variable selection, where for each covariate \( j \), there is an associated variable \( \zeta_j \) taking values in \([0, 1]\), which informs whether the variable \( j \) is important in terms of supporting a mixture distribution. Letting \( \phi_{0,j,k} \) be the observed proportion of covariate \( j \) values taking the value \( k \) throughout the whole covariate dataset \( X \), we define the new composite parameters

\[
\phi^{*}_{c,j,k} := \zeta_j \phi_{c,j,k} + (1 - \zeta_j) \phi_{0,j,k}.
\]

The likelihood model is then as in Equation 11 but with \( \phi_{Z_i,j,k} \) replaced by \( \phi^{*}_{Z_i,j,k} \). Under this model the vector \( \Lambda \) is extended by \( \zeta = (\zeta_1, \zeta_2, \ldots, \zeta_J) \). We replicate the prior structure presented in the original paper, in particular, inducing sparsity with the introduction of additional binary variables \( \rho_j \) for each \( j = 1, 2, \ldots, J \), which also extend \( \Lambda \). For extended details of the conditional posteriors and updating strategy the interested reader is referred to Papathomas et al. (2012). Of particular note is the fact that conjugacy for \( \phi_{c,j} \) is no longer retained, meaning that Metropolis-within-Gibbs updates are necessary. We use adaptive Random-walk-Metropolis proposals.

An alternative approach to the variable selection presented above, is a more traditional “hard” (binary) cluster specific variable selection. Under this model, each mixture component \( c \) has an associated vector \( \gamma_c = (\gamma_{c,1}, \gamma_{c,2}, \ldots, \gamma_{c,J}) \), where \( \gamma_{c,j} \) is a binary random variable that determines whether covariate \( j \) is important to mixture component \( c \). This allows us to write

\[
\phi^{*}_{c,j,k} := \gamma_{c,j} \phi_{c,j,k} + (1 - \gamma_{c,j}) \phi_{0,j,k} = \phi^{\gamma_{c,j}}_{c,j,k}(1 - \phi_{c,j,k})^{1-\gamma_{c,j}},
\]

which, as in the first variable selection model, is substituted into Equation 11 in place of \( \phi_{c,j,k} \), to provide the likelihood for the covariate model. Under this model, each parameter vector \( \Theta_c \) is extended by \( \gamma_c \). We use the same prior specification (and hence posterior conditionals) as the original paper, in particular introducing additional parameters \( \rho_j \) for each \( j = 1, 2, \ldots, J \) into \( \Lambda \). We also observe that using this alternative approach, the binary nature of \( \gamma_{c,j} \) means that direct Gibbs updates for \( \phi_{c,j,k} \) can still be used (albeit with slightly altered Dirichlet distributions).

Gaussian covariates

The two variable selection methods described above can equally be applied to the Gaussian mixture case. Defining \( \bar{x} = (\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_J) \), where \( \bar{x}_j = \frac{1}{n} \sum_{i=1}^n x_{i,j} \) is the average sample value of covariate \( j \), we can define the vector \( \mu^*_c = (\mu^*_{c,1}, \mu^*_{c,2}, \ldots, \mu^*_{c,J}) \) where either

\[
\mu^*_{c,j} := \zeta_j \mu_{c,j} + (1 - \zeta_j) \bar{x}_j,
\]

or

\[
\mu^*_{c,j} := \gamma_{c,j} \mu_{c,j} + (1 - \gamma_{c,j}) \bar{x}_j,
\]

depending on which variable selection approach is being adopted. We then replace \( \mu_{c,j} \) with \( \mu^*_{c,j} \) in Equation 10. Using identical priors for \( \zeta_j \) or \( \gamma_{c,j} \) and \( \rho_j \), these parameters are updated.
as for the discrete covariate case. The posterior conditional distributions for updating $\mu_c$ are shown in Appendix A.3, whereas those for updating other parameters are as before, with $\mu_{c,j}$ replaced with $\mu^*_{c,j}$ as appropriate.

4. Postprocessing of the MCMC output

The rich posterior output produced can be used to learn about the partition space and its uncertainty. It is useful to show a representative partition, as an effective way to convey the output of the clustering algorithm. Moreover, it is also of interest to assess the uncertainty associated with subgroups of this best partition.

We discuss below the necessary steps. See also Molitor et al. (2010).

1. Computation of the dissimilarity matrix. At each iteration of the sample, we can record pairwise cluster membership and construct a score matrix, with entries equal to 1 for pairs belonging to the same cluster and 0 otherwise. Averaging these matrices over the whole MCMC run leads to a similarity matrix $S$, which can be then used to identify an optimal partition.

2. Identifying the optimal partition. We have implemented two deterministic clustering procedures to characterise the optimal partition. The first is to use find the best partition by choosing the one which minimises the least-square distance to the matrix $S$. This approach is fast, but susceptible to Monte Carlo error. The second procedure implemented in the package is partitioning around medoids (PAM) on the dissimilarity matrix $1 - S$. This procedure is available in R in the package cluster and it robustly assigns individuals to clusters in a way consistent with matrix $S$. PAM is implemented for each possible number of clusters up to a specified maximum, and for each fixed number of clusters the best PAM partition is selected. A final representative cluster is then chosen by maximising the average silhouette width across these best PAM partitions.

3. Computation of the average risk and profile and the corresponding credible intervals. Given an optimal partition $P^*$ obtained as above, we examine the MCMC output to assess whether or not the model consistently clusters individuals in a manner similar to $P^*$. Consistent clustering leads to narrower credible intervals.

For example, for Bernoulli response, we obtain a distribution of the baseline risks for each cluster defined by $P^*$. At each iteration of the sampler we compute the average of baseline risks $p_{zi}$ for all individuals within a particular cluster $k$ of the optimal partition. This average baseline risk for cluster $k$ is computed as follows:

$$\bar{p}_k = \frac{1}{n_k} \sum_{i: z_i \in P^* = k} p_{zi}$$

where $n_k$ denotes the number of individuals in cluster $k$. In a similar way we can compute the distribution of cluster parameters for other response and covariates types.

5. Mixing of the MCMC algorithm
The likelihood of the DPMM is invariant to the order of cluster labels but the prior specification of the stick breaking construction is not. Therefore, to ensure adequate mixing across orderings, it is important to include label-switching moves. In this package we have implemented the two label switching moves proposed by Papaspiliopoulos and Roberts (2008) as well as a third label switching move proposed by (Hastie et al. 2013). This latter move updates the new cluster weights so they are something like their expected value conditional upon the new allocations. See (Hastie et al. 2013) for more details on the performance of this sampler.

Even with these label switching moves, convergence may be problematic and the user must address this issue using diagnostic tools. One difficulty in this respect is that there are no parameters in the model that can reliably demonstrate convergence. The parameters of the fixed effects tend to converge very quickly, regardless of the underlying clustering, as they are not cluster specific and therefore are not a good indication of the overall convergence. The cluster parameters, such as the \( \theta_c \)'s, cannot be tracked as their number (and their labels) can change from one iteration to the next. The concentration parameter \( \alpha \) is not a reliable indicator of convergence either, as discussed in Hastie et al. (2013).

To overcome this challenge, we have implemented the computation of the marginal model posterior \( p(Z|D) \) as an additional diagnostic tool. This represents the posterior distribution of the allocations given the data, having marginalised out all the other parameters (Hastie et al. 2013). The marginal model posterior is computed for each run of the MCMC and it has proved very effective for our real examples to compare runs with different initialisations and identify runs that were significantly different from others. Our experience suggests that it is harder for the MCMC algorithm to split rather than merge clusters. This means that it is important to initialise the algorithm with a number of cluster which is greater than the number of clusters that the algorithm will convergence to. The marginal model posterior can help to assess what such number is for each specific example.

Finally, while optimal partitions allow visualisation of the result of a clustering algorithm, such an approach must be applied with care as we are not aware of any effective method to quantify nor visualise clustering uncertainty. For this reason, we advise using predictions as an additional tool to assess convergence and visualise the output of the algorithm, as their posterior distributions can be compared across runs using standard methods. More details of using the package for predictions can be found in Section 7. We have observed that these posterior predictive distributions tend to be more stable than optimal partitions.

6. Software

Our implementation of the DPMM algorithm is available as an R package from CRAN. The software is primarily written in C++ and R.

The sampler implements the algorithm exactly as detailed in the current paper, although continued work is in progress to extend the scope of the software to cover additional models.

The program is further customisable through the specification of a hyperparameters, providing name-value pairs for the various hyperparameters used within the model being run. If the value is not set for a specific hyperparameter, it takes its default value. Default values can be found within the full documentation that is available as part of the software.

Moreover, this package can produce predicted values based on random allocations, or a Rao-Blackwellised estimate of predictions, where the probabilities of allocations are used instead.
of actually performing a random allocation.

7. Examples

7.1. Simulated example

We simulated 1,000 subjects, partitioned into 5 groups in a balanced manner. Ten binary
covariates were considered. To demonstrate one of the variable selection approaches within
the sampler, only the first eight covariates support a clustering structure. The response is
binary. This dataset can be simulated as follows.

R> library(PReMiuM)
R> inputs <- generateSampleDataFile(clusSummaryVarSelectBernoulliDiscrete())

We use the default values for all hyperparameters: Dirichlet conjugate priors with $a_j = 1$ for
the covariates and

$$
p(\alpha) \equiv \text{Gamma}(1, 0.5) \\
p(\theta_c) \equiv \text{t}_7(0, 2.5) \\
p(\beta) \equiv \text{t}_7(0, 2.5).
$$

We initialised all chains allocating subjects randomly to 20 groups. We run the chain for
10,000 after a burn-in sample of 20,000 iterations. While ensuring convergence is a complex
problem, we have observed good stability in all our runs, with results from independent chains
virtually identical.

R> runInfoObj <- profRegr(yModel=inputs$yModel, xModel=inputs$xModel,
nSweeps=10000, nBurn=20000, data=inputs$inputData,
output="output", covNames=inputs$covNames, nClusInit=20,
fixedEffectsNames=inputs$fixedEffectNames, run=TRUE)
R> dissimObj <- calcDissimilarityMatrix(runInfoObj)
R> clusObj <- calcOptimalClustering(dissimObj)
R> riskProfileObj <- calcAvgRiskAndProfile(clusObj)
R> clusterOrderObj <- plotRiskProfile(riskProfileObj, 'summary-sim.png')

Figure 1 shows a box-plot of the posterior distribution for the probabilities of the response
and the covariates for the 5 clusters that form the representative clustering.

7.2. Predictions

PReMiuM can produce predicted values based on simple allocations (the default), or a Rao-
Blackwellised estimate of predictions, where the probabilities of allocations are used instead
of actually performing a random allocation. The following code can be used to reproduce the
predictive distribution plotted in 2. The predictions are consistent with the simulated data.
Figure 1: Posterior distribution for the probabilities of the binary response for the representative clustering.
Figure 2: The predictive distribution for two scenarios. The covariate values for each predictive scenarios are given below each graph. ‘NA’ represents a missing value.

R> # generate data
R> inputs <- generateSampleDataFile(clusSummaryBernoulliDiscrete())
R> # set predictive scenarios
R> preds <- data.frame(matrix(c(0, 0, 1, 0, 0, 0, 0, 1, NA, 0),
ncol=5,byrow=TRUE))
R> colnames(preds) <- names(inputs$inputData)[2:
   (length(inputs$covNames)+1)]
R> # run profile regression
R> runInfoObj <- profRegr(yModel=inputs$yModel, xModel=inputs$xModel,
   nSweeps=10000, nBurn=10000, data=inputs$inputData, output="output",
   covNames=inputs$covNames, predict=preds)
R> # postprocessing
R> dissimObj <- calcDissimilarityMatrix(runInfoObj)
R> clusObj <- calcOptimalClustering(dissimObj)
R> riskProfileObj <- calcAvgRiskAndProfile(clusObj)
R> predsOutput <- calcPredictions(riskProfileObj, fullSweepPredictions=T)
R> # prediction plots
R> par(mfrow=c(1,2))
R> for (i in 1:2){
R> hist(predsOutput$predictedYPerSweep[,i,1])
R> }

7.3. Variable selection

Note that covariates 9 and 10 in Figure 1 have similar profile probabilities for all clusters, as they have been simulated not to affect the clustering. The variable selection approach will tease out the covariates that contain clustering support and exclude the others from affecting
the clustering.

We initialised the chains as in the simulated example above, with additional prior specifications given by

\[ p(\rho_p) \equiv \text{Beta}(0.5, 0.5). \]  

(13)

The algorithm consistently sampled values \( \rho_p \) in accordance with the simulated data, as shown in Fig. 3. This figure can be reproduced as follows.

\[
\begin{align*}
R & \text{> inputs } \leftarrow \text{generateSampleDataFile(clusSummaryVarSelectBernoulliDiscrete())} \\
R & \text{> runInfoObj } \leftarrow \text{profRegr(yModel=inputs$yModel, xModel=inputs$xModel, nSweeps=10000, nBurn=10000, data=inputs$inputData, output="output", covNames=inputs$covNames, varSelectType="BinaryCluster")} \\
R & \text{> rho } \leftarrow \text{summariseVarSelectRho(runInfoObj)} \\
R & \text{> par(mfrow=c(5,2))} \\
R & \text{> for (k in 1:runInfoObj$nCovariates){} } \\
R & \text{> hist(rho$rho[,k], xlab="", main="")} \\
R & \text{> } \\
\end{align*}
\]

7.4. Marginal Model Posterior

Hastie et al. (2013) introduce the marginal model posterior as a tool to assess convergence for Dirichlet process mixtures. The marginal model posterior can be computed in PReMiuM. The code below computes the marginal model posterior for four different runs of profile regression on the same dataset with different initialisations - different number of initial clusters. As it can be seen in Figure 4, plotted using the code below, for the given simulated dataset, all the MCMC runs appear to converge to subsets of the model space with equivalent marginal
model posterior. This does not imply convergence, but it is a useful diagnostic tool as it can highlight a lack of convergence in certain circumstances (Hastie et al. 2013).

```r
# generate data
R> inputs <- generateSampleDataFile(clusSummaryBernoulliDiscrete())

# select 4 different starting points
R> nClusInit<-c(10,20,50,75)

# run profile regression for all four starting points
R> for (i in 1:length(nClusInit)){
R> runInfoObj<-profRegr(yModel=inputs$yModel,
R> xModel=inputs$xModel, nSweeps=10000,
R> nBurn=10000, data=inputs$inputData,
R> output=paste("output/init",nClusInit[i],sep=""),
R> covNames = inputs$covNames, alpha=1,
R> fixedEffectsNames = inputs$fixedEffectNames,nClusInit=nClusInit[i])
R> margModelPosterior(runInfoObj)
R> }

# retrieve marginal model posterior output
R> mmp<-list()
R> for (i in 1:length(nClusInit)){
R> mmp[[i]]<-read.table(paste("output/init","nClusInit[i], 
R> _margModPost.txt",sep=""))[1]
R> }

# draw plot
R> png("plots.png")
R> plot(c(head(nClusInit,n=1)-0.5,tail(nClusInit,n=1)+0.5),
R> c(min(unlist(mmp)),max(unlist(mmp))),type="n",
R> ylab="Log marginal model posterior",
R> xlab="Initial number of clusters",
R> cex.lab=1.3,xaxt="n")
R> axis(1,at=nClusInit,labels=nClusInit)
R> for (i in 1:length(nClusInit)){
R> boxplot(mmp[[i]],add=T,at=nClusInit[i],pch=".",
R> boxwex=5,col="lightblue")
R> }
R> dev.off()
```

7.5. Run times

We have run simulations to test PReMiUM’s speed and how it scales when the number of subjects or covariates increases. The code was run in serial on an Intel(R) Xeon(R) CPU E5-2650 clocked at 2.00GHz with 20MB L3 cache, on a system with 64GB RAM.
**PReMiuM: An R Package for Profile Regression**

Figure 4: The marginal model posterior for four runs of profile regression, on the same dataset but with different initialisations.

Table 1: Time to run 100 iterations of PReMiuM for Bernoulli response and Discrete covariates

| Number of Subjects | Number of covariates | 100   | 1,000 | 10,000 |
|--------------------|----------------------|-------|-------|--------|
| 1,000              |                      | 4 sec | 1 min | 13 min |
| 2,500              |                      | 11 sec| 1.4 min| 26 min |
| 5,000              |                      | 32 sec| 3.5 min| 32 min |

**Conclusions**

The structure of PReMiuM objects gives rise to a wider variety of uses than can be described in detail here. Our intention was to provide a tutorial for Dirichlet process clustering and to illustrate the basic features of the sampler and post-processing tools that we have implemented in PReMiuM to demonstrate its utility. Our long-term goal is to continue to develop this package for analysis on complex and high dimensional datasets as well to increase the flexibility with regards to the data types that can be analysed.

**Acknowledgements**

Silvia Liverani acknowledges support from the Leverhulme Trust (ECF-2011-576). David I. Hastie acknowledges support from the INSERM grant (P27664).
Table 2: Time to run 500 iterations of PReMinM for Normal response and Normal covariates

| Number of Subjects | Number of covariates |
|--------------------|----------------------|
| 250                | 34 sec               |
| 500                | 51 sec               |
| 1,000              | 1.3 min              |

A. Appendices

A.1.

We provide the following proposition to support our assertions regarding $C^*$.

**Proposition 1.** Suppose that we have a model with posterior as given in Equation 7. Suppose $Z^*, U^*$ and $C^*$ are defined as in Section 2.2. Then:

(i) $\psi_c < U_i$ for all $i = 1, 2, \ldots, n$ and all $c > C^*$ almost surely;

(ii) $C^* \geq Z^*$ almost surely; and

(iii) $C^* < \infty$ almost surely.

**Proof.** We rely on the fact that if $V_1, V_c \sim \text{Beta}(1, \alpha)$, $\psi_1 = V_1$ and $\psi_c = V_c \prod_{l < c} (1 - V_l)$ for $c = 2, 3, \ldots$ then $\sum_{c=1}^{\infty} \psi_c = 1$. A proof of this result for the DPMM (in terms of more general conditions) is provided by Ishwaran and James (2001). Then:

(i) By definition, for all $i = 1, 2, \ldots, n$

$$U_i \geq U^* > 1 - \sum_{c=1}^{C^*} \psi_c = \sum_{c=C^*+1}^{\infty} \psi_{c'} \forall c' > C^*.$$  

(ii) Let $i^*$ be an individual $i$ such that $Z_i = Z^*$. Again, by definition, $U^* \leq U_{i^*} < \psi_{Z^*}$. This implies $1 - \psi_{Z^*} < 1 - U^*$, meaning

$$\sum_{c=1}^{Z^*-1} \psi_c < 1 - \psi_{Z^*} < 1 - U^*.$$  

By definition of $C^*$, this implies $C^* \geq Z^*$ almost surely.

(iii) Since $\sum_{c=1}^{\infty} \psi_c = 1$, this is a convergent series. By definition of a convergent series and because $U^* > 0$ we have $C^* < \infty$ almost surely.

A.2.

Below are additional comments to explain the blocking strategy employed in the DPMM algorithm. We use ‘·’ to denote “all other parameters and data”.


**Step A.** This step is a straightforward calculation of $Z^*$, which (potentially) changes at each iteration (with the update of $Z$). The set $A$ is defined immediately conditional on this value.

**Step B.** This is a joint update of $U$ and the parameters corresponding to the active components in $A$, with the inclusion of label switching moves. The principle is to use the identity $p(V^A, \Theta^A, Z, U|\cdot) = p(V^A, \Theta^A, Z|\cdot)p(U|V^A, \Theta^A, Z, \cdot)$. We proceed by first updating $(V^A, \Theta^A) \sim p(V^A, \Theta^A|Z, \cdot) = \int p(V^A, \Theta^A, U|Z, \cdot)dU$. Due to the conditional independence of $V$ and $\Theta$, this can be done in two steps: updating $V$ (B.1) then updating $\Theta$ (B.2). The moves are presented as Gibbs updates, but in fact they are Metropolis-Hastings moves, where the variable of interest (for example $V^A$) is sampled from its full conditional, and the other variables (for example $\Theta^A$ and $Z$) are kept fixed. This results in an acceptance probability of 1, making the Gibbs update equivalent. The updated values are then used as interim values for $V$ and $\Theta$ in (Metropolis-Hastings) label-switching moves (see Section 5) which are applied in B.3. The moves can change the values of $V$ as well as their order. The resulting sample of $V$ and $\Theta$ are the final updated values of these parameters for this sweep. The updated allocation vector $Z$ is used as an interim value throughout the remainder of the steps of the sweep, before the final updated value of $Z$ is sampled in Step G. The final part (B.4) of Step B is to update $U$ conditional upon the updated value of $V$ and $\Theta$ and the interim value of $Z$.

**B.1** Integrating out $U$ and taking advantage of the conjugacy of the distribution for $V$ inherent in the DPM, along with the conditional independence structure, each component of vector $V^A$ is updated by sampling $V_c \sim \text{Beta}(1 + n_c, \alpha + n_c^+)$, $c \in A$, where $n_c = \sum_i 1_{\{Z_i = c\}}$ and $n_c^+ = \sum_i 1_{\{Z_i > c\}}$.

**B.2** Integrating out $U$ and taking advantage of the conditional independence structure, $\Theta^A$ is updated from $p(\Theta^A|Z, \Theta_0, D)$. The full details of this will depend upon the application and the choice of $f$ and $P_{\Theta_0}$. Examples are given in Sections 3.

**B.3** This step implements the Metropolis-Hastings label-switching moves detailed in Section 5. These moves update $V^A$, $\Theta^A$ and $Z$ jointly from their conditional distribution with $U$ integrated out. These moves are conditional upon the values of $V^A$ and $\Theta^A$ sampled in steps B.1 and B.2.

**B.4** Conditioning on the updated values of $V^A, \Theta^A$ and $Z$ from step B.3, this step samples each $U_i$, $i = 1, \ldots, n$, independently according to the full conditional distribution, $U_i \sim \text{Unif}[0, \psi_{Z_i}] = \text{Unif}[0, V_{Z_i} \prod_{l < Z_i} (1 - V_l)]$, as detailed in Walker (2007).

**Step C.** To compute $U^*$ is straightforward given the updated value of $U$ from step B.4. The value of $Z^*$ (and with it the set $A$) can only change from that computed in Step A if the mixture component corresponding to the old $Z^*$ was involved in a label switching move, and then only if the component it was switched with was empty. By design of the label switching moves (see Section 5) this means that $Z^*$ and $A$ can only get smaller, with the consequence that parameters corresponding to a small number of components may be updated twice per MCMC sweep (once in Step B as part of the active components $A$, and once in Steps D and E as part of the updated potential components $P$). This has no ill-effects as long as the most recently updated parameter values are used at each subsequent step.
**Step D.** This is a joint update of $\alpha$, $V^P$ and $V^I$. The principle is to use the identity
\[ p(\alpha, V^P, V^I | \cdot) = p(\alpha | \cdot) p(V^P, V^I | \alpha, \cdot). \]
We proceed by first updating $\alpha \sim p(\alpha | \cdot) = \int p(\alpha, V^P, V^I | \alpha, \cdot) dV^P dV^I$ (step D.1) and then sampling $p(V^P, V^I | \alpha, \cdot)$ (step D.2). To update $V^P$, we need to alternate Gibbs samples with checks to evaluate whether the component just updated is $C^*$. In this way the set $P$ is determined on the fly. As mentioned in Section 2.1, no actual sampling is done for the inactive components in set $I$ as these would just be samples from the prior and have no impact on the likelihood or any other conditionals in the MCMC sweep.

If the prior for $\alpha$ is a Gamma distribution then it is alternatively possible to sample directly from the conditional with $V^A$ also marginalised (see Walker, 2007 and Escobar and West, 1995 for details). We retain our version as any prior for $\alpha$ can be then used.

**D.1** Since $V^P$ and $V^I$ both correspond to empty mixture components, the only contribution to the joint posterior conditional is through the prior. This allows us to easily integrate out $V^P$ and $V^I$. Due to the conditional independence, the resulting posterior from which this step samples is $p(\alpha | V^A, Z)$. Typically this cannot be sampled directly, so we employ a Metropolis-within-Gibbs move to update $\alpha$, using an adaptive random-walk-Metropolis proposal on the log-scale.

**D.2** We begin by setting $C = Z^*$. We then repeat the following two steps until the stop condition is reached. First, check if $\sum_{c=1}^{C} \psi_c > 1 - U^*$. Next, if the condition is met we set $C^* = C + 1$ and sample $V_C \sim \text{Beta}(1, \alpha)$.

**Step E.** This step updates the parameters $\Theta^P$ and $\Theta^I$ from the distribution $p(\Theta^P, \Theta^I | \cdot)$. The set $P$ is fully determined from step D.2. The parameters correspond to empty mixture components, so updated values of $\Theta^P$ are sampled directly from the prior. As with other inactive parameter $\Theta^I$ play no part in this MCMC sweep and so are not updated.

**E.1** Taking advantage of the conditional independence structure, in this step we update $\Theta^P$ by doing a Gibbs sample from the prior, such that $\Theta_c \sim p(\Theta_c | \Theta_0)$ for each $c \in P$. As $\Theta_c$ may be a vector of parameters, this may involve a number of Gibbs updates per component $c$. The full details depend upon the choice of $P_{\Theta_0}$. See Section 3 for examples.

**Step F.** Here, the global (non-cluster-specific) likelihood parameters $\Lambda$ associated with $f$ are updated. There are only a finite number of such parameters so no special updates are needed.

**F.1** Sample $\Lambda \sim p(\Lambda | \Theta^A, Z, D)$. Due to the conditional independence structure of the model the update only depends on the current value of the active likelihood parameters $\Theta^A$, the allocations $Z$, and the data $D$. $\Lambda$ may contain multiple parameters, so this stage may contain many Gibbs and / or Metropolis-within-Gibbs steps. Full details will depend upon the choice of $f$ and $p(\Lambda)$, see Section 3 for examples.

**Step G.** The final step of the algorithm is to update the parameter allocations $Z$, conditional on the newly updated values of the other parameters.
G.1 We sample $Z \sim p(Z | V^A, V^P, \Theta^A, \Theta^P, U, \Lambda, D)$. Because of the independence of the individuals $i$, this is a series of Gibbs updates for each $i = 1, 2, \ldots, n$, sampling $Z_i \sim p(Z_i = c | V^A, V^P, \Theta^A, \Theta^P, U_i, \Lambda, D_i)$ where $D_i$ is the data for individual $i$. For each update, since the conditional $p(Z_i = c | \cdot)$ has no posterior mass for clusters $c$ where $U_i > \psi_c$, this update depends only on the parameters associated with the finite number of clusters in the sets $A$ and $P$, making the update a simple multinomial sample according to a finite vector of weights. The full details of the weights will depend upon the choice of $f$ and $P_{\Theta_0}$.

A.3.

In the case of variable selection for continuous covariates, define the $J \times J$ matrix $\Gamma_c$ as

$$
\Gamma_{c,i,j} = \begin{cases} 
\gamma_{c,j} & i = j; \quad j = 1, 2, \ldots, J \\
0 & \text{otherwise},
\end{cases}
$$

for variable selection method 1, and

$$
\Gamma_{c,i,j} = \begin{cases} 
\zeta_j & i = j; \quad j = 1, 2, \ldots, J \\
0 & \text{otherwise}.
\end{cases}
$$

for method 2 as presented in Section 3.5.2. Let $I_J$ denote the $J \times J$ identity matrix, $n_c$ be the number of individuals allocated to cluster $c$, $\bar{X}$ be as defined in Section 3.5.2 and $\bar{X}_c = (\bar{X}_{c,1}, \bar{X}_{c,2}, \ldots, \bar{X}_{c,J})$ such that $\bar{X}_{c,j} = \sum_{i: Z_i = c} X_{i,j}/n_c$. The posterior conditional distributions for updating $\mu_c$ for $c \in A$ are then given by

$$
\mu \sim \text{Normal}(\tilde{\mu}, \tilde{\Sigma})
$$

where

$$
\tilde{\Sigma} = (\Sigma_0^{-1} + n_c \Gamma \Sigma_c^{-1} \Gamma)^{-1},
$$

and

$$
\tilde{\mu} = \tilde{\Sigma} [\Sigma_0^{-1} \mu_0 + n_c \Gamma \Sigma_c^{-1} (\bar{X}_c - (I_J - \Gamma) \bar{X})].
$$

References

Andrieu C, Thoms J (2008). “A tutorial on adaptive MCMC.” *Statistics and Computing*, 18(4), 343–373.

Bigelow JL, Dunson DB (2009). “Bayesian Semiparametric Joint Models for Functional Predictors.” *Journal of the American Statistical Association*, 104(485), 26–36.

Blackwell D, MacQueen JB (1973). “Ferguson distributions via Polya Urn Schemes.” *Annals of Statistics*, 1(2), 353–355.

Chung Y, Dunson DB (2009). “Nonparametric Bayes conditional distribution modeling with variable selection.” *Journal of the American Statistical Association*, 104(488), 1646–1660.
Dunson DB (2009). “Nonparametric Bayes local partition models for random effects.” *Biometrika, 96*(2), 249–262.

Dunson DB, Herring AB, Siega-Riz AM (2008). “Bayesian Inference on Changes in Response Densities Over Predictor Clusters.” *Journal of the American Statistical Association, 103*(484), 1508–1517.

Escobar MD, West M (1995). “Bayesian density estimation and inference using mixtures.” *Journal of the American Statistical Association, 90*(430), 577–588.

Ferguson TS (1973). “A Bayesian analysis of some nonparametric problems.” *Annals of Statistics, 1*(2), 209–230.

Green PJ (2010). “Colouring and breaking sticks: random distributions and heterogeneous clustering.” In NH Bingham, CM Goldie (eds.), *Probability and Mathematical Genetics: Papers in Honour of Sir John Kingman*, pp. 319–344. Cambridge University Press, Cambridge, UK.

Hastie DI, Liverani S, Richardson S (2013). “Sampling from Dirichlet process mixture models with unknown concentration parameter.” Submitted.

Ishwaran H, James LF (2001). “Gibbs sampling methods for stick-breaking priors.” *Journal of the American Statistical Association, 96*(453), 161–173.

Kalli M, Griffin JE, Walker SG (2011). “Slice sampling mixture models.” *Statistics and Computing, 21*(1), 93–105.

Moñitor J, Papathomas M, Jerrett M, Richardson S (2010). “Bayesian profile regression with an application to the National Survey of Children’s Health.” *Biostatistics, 11*(3), 484–498.

Moñitor J, Su JG, Moñitor NT, Rubio VG, Richardson S, Hastie D, Morello-Frosch R, Jerrett M (2011). “Identifying vulnerable populations through an examination of the association between multipollutant profiles and poverty.” *Environmental Science & Technology, 45*(18), 7754–7760.

Neal RM (2000). “Markov chain sampling methods for Dirichlet process mixture models.” *Journal of Computational and Graphical Statistics, 9*(2), 249.

Papaspiliopoulos O (2008). “A note on posterior sampling from Dirichlet mixture models.” *Technical Report 8*, CRISM Paper.

Papaspiliopoulos O, Roberts GO (2008). “Retrospective Markov chain Monte Carlo methods for Dirichlet process hierarchical models.” *Biometrika, 95*(1), 169–186.

Papathomas M, Moñitor J, Hoggart C, Hastie DI, Richardson S (2012). “Exploring data from genetic association studies using Bayesian variable selection and the Dirichlet process: application to searching for gene × gene patterns.” *Genetic Epidemiology, 6*(36), 663–74.

Papathomas M, Moñitor J, Richardson S, Riboli E, Vines P (2011). “Examining the joint effect of multiple risk factors using exposure risk profiles: lung cancer in non-smokers.” *Environmental Health Perspectives, 119*, 84–91.
Pitman J, Yor M (1997). “The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator.” *Annals of Probability, 25*(2), 855–900.

R Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Rodriguez A, Dunson DB (2011). “Nonparametric Bayesian models through probit stick-breaking processes.” *Bayesian Analysis, 6*(1), 145–178.

Sethuraman J (1994). “A constructive definition of Dirichlet priors.” *Statistica Sinica, 4*, 639–650.

Walker SG (2007). “Sampling the Dirichlet mixture model with slices.” *Communications in Statistics - Simulation and Computation, 36*, 45–54.

Yau C, Papaspiliopoulos O, Roberts GO, Holmes C (2011). “Bayesian non-parametric hidden Markov models with applications in genomics.” *Journal of the Royal Statistical Society, Series B (Statistical Methodology), 73*, 37–57.

**Affiliation:**

Sylvia Richardson  
MRC Biostatistics Unit  
Cambridge, UK  
E-mail: sylvia.richardson@mrc-bsu.cam.ac.uk