Gibbs sampling methods for Pitman-Yor mixture models

Mame Diarra Fall, Éric Barat

Gibbs Sampling Methods for Pitman-Yor Mixture Models

Mame Diarra Fall[1] and Éric Barat[2]

[1] MAPMO - UMR 6628 Fédération Denis Poisson
Université d'Orléans 45067 Orléans cedex 2, France

[2] Laboratoire de Modélisation, Simulation et Systèmes
CEA/DRT/LIST/DCSI/LM2S, 91191 Gif-sur-Yvette, France.

e-mail: diarra.fall@univ-orleans.fr ; eric.barat@cea.fr

Abstract

We introduce a new sampling strategy for the two-parameter Poisson-Dirichlet process mixture model, also known as Pitman-Yor process mixture model (PYM). Our sampler is therefore applicable to the well-known Dirichlet process mixture model (DPM).

Inference in DPM and PYM is usually performed via Markov Chain Monte Carlo (MCMC) methods, specifically the Gibbs sampler. These sampling methods are usually divided in two classes: marginal and conditional algorithms. Each method has its merits and limitations. The aim of this paper is to propose a new sampler that combines the main advantages of each class. The key idea of the proposed sampler consists in replacing the standard posterior updating of the mixing measure based on the stick-breaking representation, with a posterior updating of [Pit96b] which represents the posterior law under a Pitman-Yor process as the sum of a jump part and a continuous one. We sample the continuous part in two ways, leading to two variants of the proposed sampler. We also propose a threshold to improve mixing in the first variant of our algorithm.

The two variants of our sampler are compared with a marginal method, that is the celebrated Algorithm 8 of [Nea00], and two conditional algorithms based on the stick-breaking representation, namely the efficient slice sampler of [KGW11] and the truncated blocked Gibbs sampler of [IJ01]. We also investigate effects of removing the proposed threshold in the first variant of our algorithm and introducing the threshold in the efficient slice sampler of [KGW11]. Results on real and simulated data sets illustrate that our algorithms outperform the other conditionals in terms of mixing properties.

Keywords: Bayesian nonparametrics; Dirichlet process mixture model; Pitman-Yor process mixture model; Gibbs sampler; Slice sampling

1 Introduction

Bayesian nonparametrics have recently gained popularity in a great number of applications in statistics and machine learning (density estimation, clustering, image segmentation and reconstruction, language modeling etc.). Dirichlet process mixture models (DPM) ([Fer83], [Lo84]) have become ubiquitous in Bayesian nonparametric modeling as reviewed by [MQ04] and recently by [MM13]. This makes crucial the use of effective sampling strategies for DPM and their two-parameter generalization a.k.a as Pitman-Yor mixtures (PYM). In particular, the mixing properties of MCMC samplers appears as a key point in order to address high dimension applications using large datasets. In this paper, we investigate such a task.

A DPM assumes that the random density function can be written as

$$f(\mathbf{x}) = \int p(\mathbf{x}|\boldsymbol{\theta})\mathrm{d}H(\boldsymbol{\theta}) \quad \text{with} \quad H \sim \mathrm{DP}(\alpha, G_0), \tag{1}$$

where $\{p(\cdot|\boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta\}$ is a family of non-negative (possibly multivariate) kernels defined on a complete and separable metric space $\mathcal{X}$ such that $\int_{\mathcal{X}} p(\mathbf{x}|\boldsymbol{\theta})\lambda(\mathrm{d}\mathbf{x}) = 1$ for all $\boldsymbol{\theta} \in \Theta$ and for some σ-finite measure λ. The prior over the mixing distribution H is a Dirichlet process (DP) [Fer73] with parameters $\alpha > 0$ and base distribution G_0 that is the prior guess at the shape of H, $\mathbb{E}[H(\cdot)] = G_0(\cdot)$. As it is well-known, the DP selects an almost surely discrete random probability measure that can be represented as

$$H(\cdot) = \sum_{k=1}^{\infty} w_k \delta_{\boldsymbol{\theta}_k^*}(\cdot), \tag{2}$$

where $(w_k)_{k=1}^{\infty}$ are non-negative weights that sum to unity, and $(\boldsymbol{\theta}_k^*)_{k=1}^{\infty}$ a sequence of Θ-valued random locations. The random probability measure H can be constructed using the stick-breaking representation, where the weights are

$$w_k = v_k \prod_{l<k}(1 - v_l)$$

for a sequence $(v_k)_{k\geq 1}$ of independent random variables on $(0, 1)$. One can replace the Dirichlet process in (1) with any almost surely discrete random probability measure, for example a Pitman-Yor process (PYP). The definition of the PYP is set forth in the next paragraph, but here we anticipate that the Pitman-Yor process mixture model is obtained as follows

$$f(\mathbf{x}) = \int p(\mathbf{x}|\boldsymbol{\theta})\mathrm{d}H(\boldsymbol{\theta}) \quad \text{with} \quad H \sim \mathrm{PY}(d, \alpha, G_0), \tag{3}$$

with $d \in [0, 1)$, $\alpha > -d$. The PYM is an interesting alternative to the DPM that allows more flexibility in the modeling. Alternatively, the model (3) can be expressed hierarchically as follows

$$\begin{aligned} \mathbf{x}_i|\boldsymbol{\theta}_i &\overset{ind}{\sim} p(\mathbf{x}_i|\boldsymbol{\theta}_i) \\ \boldsymbol{\theta}_i|H &\overset{iid}{\sim} H \\ H|\alpha, G_0 &\sim \mathrm{PY}(d, \alpha, G_0). \end{aligned} \tag{4}$$

By exploiting the discreteness of H, the PYM also provides a flexible model for clustering items of various kinds in a hierarchical setting without explicitly specifying the number of components.

In a Bayesian context, one is interested in the posterior distribution of the random density f. However, this later does not exhibit any closed form and inference is necessarily simulation-based. After the first MCMC method introduced by [Esc94] for DP mixture models, many authors contributed to its improvement ([EW95], [MM98]) and to handle non-conjugate cases [Nea00]. These aforementioned techniques integrate with respect to the mixing measure H and, thus, remove the infinite dimensional aspect of the problem. This leads to the so-called marginal (or Pólya urn) approaches. Efficient versions of marginal samplers usually achieve good mixing performances [Nea00]. An alternative to marginal methods is given by the conditional algorithms which explicitly represent the mixing measure using, for instance, its stick-breaking series representation ([IJ01], [Wal07]). The challenge in conditional approaches is to deal with the countably infinite representation of H in Equation (2). In [IJ01], authors resort to an approximation and truncate the mixing measure at a deterministic value. An alternative which avoids hard truncations was proposed by [MT98] who provided an approximation of the Dirichlet process by means of a random truncation of its stick-breaking representation. The idea of random truncation has also been developed in [PR07] with a Metropolis-Hastings sampling scheme, and in [Wal07] using the slice sampling strategy. This later algorithm has been improved by [Pap08] and [KGW11].

The key advantage of conditional methods using the stick-breaking construction is in updating the mixing measure H as well as the other parameters in the model. This makes possible direct inference on H. Furthermore, components weights are explicitly represented and updated. This property makes these algorithms able to update blocks of parameters and easy to parallelize in order to take advantage of recent parallel computation hardware architectures, which is well suited particularly for large data sets. On the flip side, by integrating mixture components out of the model, marginal techniques are based on incremental updates which are prejudicial when working with huge data sets. Another drawback of marginalizing over the mixing measure is that computing posterior conditionals require additional sampling steps (see [IJ01]).

However, dealing with exchangeable prediction rules, marginal methods exhibit most of the time better mixing properties than conditionals, and our experimental comparison in Section 5 corroborates this assessment. Also, the random weights are collapsed by marginalization and this results in a crucial reduction of the parameters space dimension. A limitation of conditional methods based on the stick-breaking representation is that the sampler operates in the space of non-exchangeable cluster labels, as pointed out in the paper by [PISW06]. Indeed, in this representation, weights are explicitly defined by the prior and components are represented with a size-biased ordering over their labels. This means that components with lower labels have higher prior probabilities than components with higher labels. As a consequence, components are not interchangeable and cluster prior labelling contributes to the posterior sampling. In this situation, the sampler needs to mix efficiently over clusters labels to avoid any clustering bias. Then, [PISW06] recommend systematic use of two additional Metropolis-Hastings moves ("label-swap" and "label-permute") in order to improve mixing over clusters. When working with non-exchangeable clusters

labels, this additional step seems to be the only way to improve the mixing over clusters (see also [Pap08]). In contrast, in marginal methods using Pólya urn representation, the sampling occurs in the space of equivalence classes over exchangeable clusters labels where clusters identities are arbitrary and insignificant. This is the adequate space to live for the sampler because cluster labels are irrelevant.

In this context, we introduce a new conditional sampling scheme which is formulated in the space of equivalence classes over clusters labels where clusters identities are irrelevant. Our approach can be seen as marginalizing the clusters labels ordering. Before going further, we point out that instead of using the stick-breaking representation for the underlying mixing measure, recent conditional samplers exploit other constructive representations. For instance, the use of the so-called Ferguson and Klass representation [FK72] of independent increment processes has been recently considered in the literature. See, e.g., [GW11] and [NBP09], and references therein for some recent contributions in this direction. Such approaches are interesting in the sense that they allow to consider classes of priors, in general, wider than the PYP and the DP. However, they become non-trivial to implement, even when applied to the DP model. Given the importance of the DPM and PYM, the dominant priors in Bayesian nonparametrics, it seems important to devote attention to the development of alternative, simple and efficient algorithms. It is the purpose of this paper to provide a simple and fast way to infer DPM and PYM. Also, it is worth mentioning that since this work was presented in a technical report, it has been successfully applied, for example in [CTM].

The remainder of this paper is structured as follows. In Section 2, we recall some preliminaries about Pitman-Yor processes. Section 3 briefly recalls the basis of the algorithms that are used to compare our samplers and discuss the advantages and limitations of marginal and conditional methods. Afterwards, in Section 4, we present the two variants of the proposed MCMC. We evaluate performance of the algorithms through application to real and simulated data sets in Section 5. Finally, we conclude the paper in Section 6 with discussions and extensions for further work.

2 Preliminaries on Pitman-Yor processes

In this section, we provide a succinct description of the Pitman-Yor process. We refer interested readers to [Pit96b], [PY97], [Pit02],[IJ01], [IJ03] and references therein for more details on Pitman-Yor processes.

Definition 1 *(Two-parameter GEM distribution)*
Let $d \in [0,1)$, $\alpha > -d$, $(v_j)_{j\geq 1}$ a sequence of independent random variables such that for all j, $v_j \sim Beta(1-d, \alpha + jd)$. Define the sequence of weights (w_j) by the stick-breaking scheme as follows $w_1 = v_1, w_2 = v_2(1-v_1), \ldots, w_j = v_j \prod_{i=1}^{j-1}(1-v_i)$. The sequence $\mathbf{w} = (w_j)_{j\geq 1}$ is said to follow a two-parameter GEM distribution (after Griffiths, Engen and McCloskey), with parameters d and α, and denoted by

$$\mathbf{w} \sim GEM(d, \alpha).$$

Definition 2 *(Pitman-Yor process)*
Let $\mathbf{w} \sim GEM(d, \alpha)$ and G_0 a diffuse (non-atomic) probability measure on a

measurable space $(\Theta, \mathcal{B})$ *(i.e* $G_0(\{\theta\}) = 0$ *for each* $\theta \in \Theta$*). Let* $\mathbf{\Theta}^* = (\boldsymbol{\theta}_j^*)_{j\geq 1}$ *be iid* G_0*, independently of* $\mathbf{w}$*. Then*

$$H(\cdot) \stackrel{d}{=} \sum_{k=1}^{\infty} w_k \delta_{\boldsymbol{\theta}_k^*}(\cdot), \tag{5}$$

where $\delta_{\boldsymbol{\theta}^*}(\cdot)$ *denotes the Dirac measure giving mass* 1 *at* $\boldsymbol{\theta}^*$*, is distributed according to a two-parameter Poisson-Dirichlet distribution on* $(\Theta, \mathcal{B})$*. We refer to* H *as a Pitman-Yor process with parameters* d *and* α *and base measure* G_0*. It is denoted by:*

$$H \sim PY(d, \alpha, G_0).$$

Setting $d = 0$, the Pitman-Yor process reduces to the Dirichlet process with parameters α and G_0, while the PY$(d, 0, G_0)$ yields a measure whose random weights are based on a stable law with index $0 < d < 1$.

Definition 2 is a constructive definition of the PYP called *stick-breaking* representation. The PYP has also a characterization in terms of generalized Blackwell-MacQueen [BM73] urn scheme. Henceforth, K_n will denote the random variable identifying the number of distinct values appearing in the sample $(\boldsymbol{\theta}_1 \ldots, \boldsymbol{\theta}_n)$ from a two-parameter Poisson-Dirichlet process, and k_n a realization of K_n.

Proposition 1 *(Generalized Blackwell-MacQueen urn)*
Let $d \in [0, 1)$*,* $\alpha > -d$ *and* G_0 *a diffuse probability measure on* Θ*. Consider a sequence of* $(\boldsymbol{\theta}_i)_{i\geq 1}$ *generated via the following predictive distributions*

$$\boldsymbol{\theta}_1 \sim G_0 \tag{6}$$

$$\boldsymbol{\theta}_{n+1} | \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n \sim \frac{\alpha + dk_n}{\alpha + n} G_0 + \sum_{j=1}^{k_n} \frac{n_j - d}{\alpha + n} \delta_{\boldsymbol{\theta}_j^*}, \tag{7}$$

where $\{\boldsymbol{\theta}_j^*, j = 1, \ldots, k_n\}$ *are the unique values among* $\{\boldsymbol{\theta}_i, i = 1, \ldots, n\}$ *and* n_j *the frequency of* $\boldsymbol{\theta}_j^*$*. The distribution of this sequence of exchangeable draws converges almost surely to a discrete distribution which is distributed according to a* $PY(d, \alpha, G_0)$ *when* n *goes to infinity:* $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n | H \sim H$ *where* $H \sim PY(d, \alpha, G_0)$*.*

The predictive distributions (6)-(7) are the key components of marginal methods described in Section 3. They also make clear the role of the parameters in Pitman-Yor and Dirichlet process mixture models. In a DPM, the probability for $\boldsymbol{\theta}_{n+1}$ to coincide with an already observed value, say $\boldsymbol{\theta}_j^*$, is determined by n_j while the probability to sample a new value for $\boldsymbol{\theta}_{n+1}$ from the base measure G_0 is proportional to the concentration parameter α. As a consequence, α is the only parameter which can be used to tune the number of clusters. [KH73] showed that the asymptotic behaviour of the number of clusters K_n that are induced by the Dirichlet process is

$$K_n / \log(n) \to \alpha.$$

For some applications, this feature can be too restrictive. In a PYM, together α and d control the formation of new clusters. The first controls the overall

number of clusters whereas d determines the asymptotic growth of K_n. The larger d, the more new values are generated. In [Pit02], it is shown that

$$K_n/n^d \to S_d$$

where S_d is a strictly positive random variable, with continuous density

$$\mathbb{P}_{d,\alpha}(S \in ds) = g_{d,\alpha}(s) := \frac{\Gamma(\alpha+1)}{\Gamma(\frac{\alpha}{d}+1)} s^{\frac{\alpha}{d}} g_d(s) ds \quad (s > 0)$$

where $g_d = g_{d,0}$ is the density function of a positive stable random variable with parameter d.
These effects can be visualized by looking at the induced prior distribution of the number of distinct clusters in a sample of size running from 1 to $10,000$, for both DP and PYP with respectively $\alpha = 10$ and $(d = 0.5, \alpha = 10)$. Results are displayed in Figure 1.

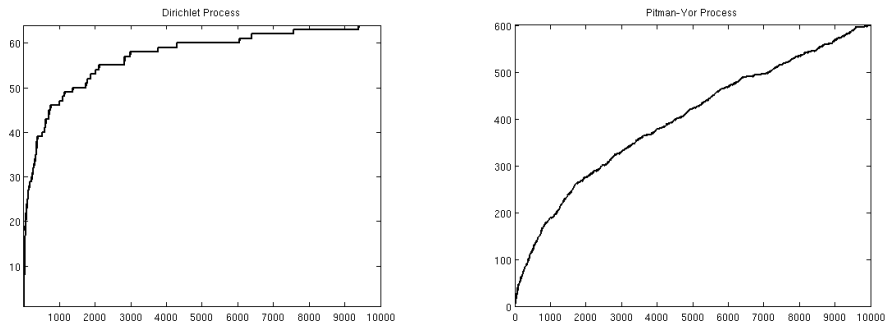

Figure 1: Number of clusters as a function of the sample size for a DP($\alpha = 10$) (left) and a PYP($d = 0.5, \alpha = 10$) (right). The rate at which the number of clusters K_n increases is slower in the DP than in the PYP, being respectively $\log(n)$ and n^d.

Another characteristic of the DP and the PYP is a reinforcement mechanism (named after "rich-get-richer") that tends to reinforce among the observed clusters those having higher frequencies. As noted above, in a DP the probability of joining an existing cluster j is proportional to the size n_j of that cluster. In a PYP however, this probability is proportional to $n_j - d$, with d acting as a discount parameter that reduces the probability of adding a new observation to an existing cluster. This yields a power-law behaviour for the PYP which greatly influences the clustering structure and makes this prior well-suited to natural language processing applications, unlike the DP [Teh06]. We refer to [DBFL$^+$] for more details on the role of d in the combined effect of the reinforcement mechanism and the increase in the rate at which new values are generated.

After having recalled the basis of Pitman-Yor processes, we present in the next section the MCMC samplers of both marginal and conditional type for the Dirichlet and Pitman-Yor mixture models.

3 Sampling from DPM and PYM

To be self-contained, in this section we recall basis of the two classes of MCMC algorithms for sampling under the posterior of a DPM and a PYM and briefly describe each of the algorithms that are used to compare our proposed method.

3.1 MCMC algorithms for PYM

Posterior distributions are intractable in DPM and PYM. Posterior inference is performed using approximation techniques such MCMC methods. There are many sampling MCMC algorithms which can roughly be divided into two categories: marginals and conditionals.

3.1.1 Marginal methods

These methods are called marginal since the infinite dimensional random component, namely the mixing measure H, is integrated out of the model and the predictive distributions are used within a Gibbs sampler to get posterior samples. Marginal methods can be sub-categorized into conjugate or non-conjugate models. By conjugacy, we mean that the mixture kernel $p(\cdot|\boldsymbol{\theta})$ and the base distribution G_0 form a conjugate pair. In this case, calculations in conditional posterior distributions are simplified and can be performed analytically ([Nea91], [Esc94], [WME94], [EW95] and [BM96]). In non-conjugate models however, posteriors can not be easily calculated. The sampling scheme is more difficult and requires elaborated techniques ([MM98], [WD98] and [GR01]). The reader is referred to [Nea00] for a more complete overview and discussions about these methods. Neal [Nea00] also proposes two novel sampling schemes for non-conjugate models: the first (referred to as "algorithm 7" in Neal's paper) uses a combination of Metropolis-Hastings steps with Gibbs updates. The second, named after "algorithm 8", is based on an augmentation scheme and extends the model to include auxiliary components which exist temporarily. We briefly detail this algorithm we will use to contrast our sampler since, to our knowledge, it achieves the best mixing properties in marginal methods. This algorithm has been developed for Dirichlet process mixture models in Neal's paper. Here, we slightly modify it by adding the second parameter in order to infer Pitman-Yor mixture models.

Algorithm 8 of [Nea00]: The idea behind "algorithm 8" of [Nea00] is to add auxiliary components (representing potential future components) when updating classification variables.

To be more precise, let c_i such that $c_i = k$ iff $\boldsymbol{\theta}_i = \boldsymbol{\theta}^*_k$, where $\boldsymbol{\Theta}^* = (\boldsymbol{\theta}^*_1, \boldsymbol{\theta}^*_2, \dots)$ denote the unique values among $(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_n)$. Since data are exchangeable and labels of components completely arbitrary, each datum $\mathbf{x}_i$ can be treated as the last. Using the predictive distribution (7), $\mathbf{x}_i$ is thus assigned to an already represented component or to an auxiliary component. If we denote by k^-_n the number of active components disregarding observation i and $n_{-i,k}$ the number of c_j for $j \neq i$ that are equal to k, the prior probability to allocate $\mathbf{x}_i$ to an active component is $\frac{n_{-i,k}-d}{\alpha+n-1}$ and the probability to create a new component is $\frac{\alpha+dk^-_n}{\alpha+n-1}$, which will be equally distributed among the m auxiliary components.

The choice of m is left to the user. It is governed by a balance between computational considerations and mixing properties. Combining with the likelihood, updating classification variables is done via the conditional probabilities:

$$\Pr(c_i = k|\mathbf{x}_i, \mathbf{c}_{-i}, \boldsymbol{\Theta}^*) \propto \begin{cases} \dfrac{n_{-i,k} - d}{n - 1 + \alpha}\ p(\mathbf{x}_i|\boldsymbol{\theta}_k^*) \text{ for } k = 1, \ldots, k_n^-, \\ \dfrac{\alpha + dk_n^-}{m(n - 1 + \alpha)}\ p(\mathbf{x}_i|\boldsymbol{\theta}_k^*) \text{ for } k = k_n^- + 1, \ldots, k_n^- + m. \end{cases}$$

After this step, parameters for non-empty components are updated according to their posterior law based on the prior G_0 and the likelihood of all data currently allocated to:

$$p(\boldsymbol{\theta}_k^*|\mathbf{X}, \mathbf{c}) \propto G_0(d\boldsymbol{\theta}_k^*) \prod_{\{i:c_i=k\}} p(\mathbf{x}_i|\boldsymbol{\theta}_k^*).$$

3.1.2 Conditional methods

In contrast to marginal methods, conditional samplers retain the mixing measure H and explicitly represent it using, for example, the stick-breaking construction stated in Equation (5). The issue is to treat the infinite dimensionality of H. There are methods that approximate H by a deterministic truncation of the number of its components, and those that use a finite but random number of masses.

The truncated blocked Gibbs sampler [IJ01]

To remove the infinite dimensional aspect of the problem, Ishwaran and James ([IJ01], [IJ03]) truncate H at a chosen integer value N. It is necessary to set $v_N = 1$ in the stick-breaking construction of H to ensure that the truncated measure H_N is a probability measure, with distribution:

$$H_N(\cdot) = \sum_{k=1}^{N} w_k \delta_{\boldsymbol{\theta}_k^*}(\cdot).$$

Under this truncated framework, and using classification variables $\mathbf{c} = (c_1, \ldots, c_n)$, the hierarchical model (4) can be rewritten as follows

$$\begin{aligned} \mathbf{x}_i|c_i, \boldsymbol{\Theta}^* &\sim p(\mathbf{x}_i|\boldsymbol{\theta}_{c_i}^*) \\ c_i|\mathbf{w} &\sim \sum_{k=1}^{N} w_k \delta_k(\cdot) \\ \mathbf{w}|d, \alpha &\sim \text{GEM}(d, \alpha) \\ \boldsymbol{\theta}_k^*|G_0 &\sim G_0, \end{aligned} \tag{8}$$

where $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ represent the data, $\boldsymbol{\Theta}^* = (\boldsymbol{\theta}_1^*, \ldots, \boldsymbol{\theta}_N^*)$ the parameters of the mixture components and $\mathbf{w} = (w_1, \ldots, w_N)$ the components weights, with $w_N = \prod_{k=1}^{N-1}(1 - v_k) = 1 - \sum_{k=1}^{N-1} w_k$. Rewriting the model in this form makes direct posterior inference possible since one has to treat a finite number of components. If we denote by k_n the number of currently non-empty components, the full conditionals involved in the Gibbs sampler are given by the following:

1. Conditional for $\mathbf{c}$: $c_i|\mathbf{w}, \mathbf{\Theta}^*, \mathbf{X} \sim \sum_{k=1}^{N} w_{k,i}\delta_k$ for $i = 1, \ldots, n$ where $(w_{1,i}, \ldots, w_{N,i}) \propto (w_1\ p(\mathbf{x}_i|\boldsymbol{\theta}_1^*), \ldots, w_N\ p(\mathbf{x}_i|\boldsymbol{\theta}_N^*))$.

2. Conditional for $\mathbf{w} : w_1 = v_1^*$ and $w_k = v_k^* \prod_{l=1}^{k-1}(1 - v_l^*)$ for $k = 2, \ldots, N-1$, with $v_k^* \sim \text{Beta}(1 - d + n_k,\ \alpha + kd + \sum_{l=k+1}^{N} n_l)$ for $k = 1, \ldots, N-1$, where $n_k = \#\{i : c_i = k\}$.

3. Conditional for $\mathbf{\Theta}^*$:

   - $\boldsymbol{\theta}_k^* \sim G_0$ for $k = k_n + 1, \cdots, N$.
   - $\boldsymbol{\theta}_k^*|\mathbf{c}, \mathbf{X}$ has density proportional to

$$G_0(\mathrm{d}\boldsymbol{\theta}_k^*) \prod_{\{i:c_i=k\}} p(\mathbf{x}_i|\boldsymbol{\theta}_k^*) \quad \text{for } k = 1, \ldots, k_n.$$

This sampler is easy to implement since the truncation allows it to be similar to standard Gibbs samplers in finite dimensional models. However, even if methods for controlling the truncation accuracy have been proposed ([IJ01], [IJ03]), it would be better to avoid any hard approximation. To this purpose, algorithms that only use a finite number of elements at any iteration while allowing inference to the true infinite-dimensional prior have been proposed by Papaspiliopoulos and Roberts [PR07] and Walker [Wal07]. This later uses an elegant strategy called *slice sampling* and is based on auxiliary variables. Walker's slice sampling was improved by Papaspiliopoulos [Pap08] and Kalli *et al.* [KGW11]. This later is named after "slice efficient".

The slice sampler ([Wal07], [KGW11])

The idea under the slice sampler proposed by [Wal07] for inference in DPM is to introduce uniform auxiliary variables which make the mixture model conditionally finite. To make it precise, let us consider the model (4) with kernel p and mixing measure H being constructed using the stick-breaking representation (5). The density of a single observation $\mathbf{x}_i$, given $\mathbf{w}$ and $\mathbf{\Theta}^*$, is

$$f(\mathbf{x}_i) = \sum_{k=1}^{\infty} w_k\ p(\mathbf{x}_i|\boldsymbol{\theta}_k^*). \tag{9}$$

The infinite dimensional aspect of (9) is tackled by introducing $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ uniform auxiliary variables such that the joint density of any $(\mathbf{x}_i, u_i)$ is

$$f(\mathbf{x}_i, u_i) = \sum_{k=1}^{\infty} \mathbf{1}(u_i < w_k)\ p(\mathbf{x}_i|\boldsymbol{\theta}_k^*) = \sum_{k=1}^{\infty} w_k\ p(\mathbf{x}_i|\boldsymbol{\theta}_k^*)\,\mathcal{U}(u_i|0, w_k),$$

where $\mathcal{U}(\cdot|a, b)$ denotes the uniform density function over $[a, b]$. The conditional density of $\mathbf{x}_i$, given u_i, is

$$f(\mathbf{x}_i|u_i) = \frac{1}{\sum_{k=1}^{\infty} \mathbf{1}(u_i < w_k)} \sum_{k \in \{j:w_j > u_i\}} p(\mathbf{x}_i|\boldsymbol{\theta}_k^*).$$

So, given u_i, the number of components of the mixture model f is finite. One can then complete the model by introducing a further assignment variable c_i and considering the joint density function

$$f(\mathbf{x}_i, u_i, c_i) = \mathbf{1}(u_i < w_{c_i})\ p\left(\mathbf{x}_i|\boldsymbol{\theta}_{c_i}^*\right) = w_{c_i}\ \mathcal{U}(u_i|0, w_{c_i})\ p\left(\mathbf{x}_i|\boldsymbol{\theta}_{c_i}^*\right).$$

The full conditionals required to implement this Gibbs sampler are those of: the slice variables u_i, the indicators c_i, the components parameters $\boldsymbol{\theta}_k^*$ and the stick-breaking weights v_k. In the slice sampler of Walker [Wal07], each of these variables is sampled independently and the sampling of the v_k is quite hard. This was handled in the efficient version of the slice sampler ("slice efficient") proposed in [KGW11]. In this version, the stick weights and the slice variables are blocked during iterations, which, by integrating out slice variables, dramatically simplifies generation of the weights and results in a more efficient sampler compared to Walker's algorithm. So, we will compare our methods with the "slice efficient" one. The required full conditionals of this later algorithm are now given.

1. Conditional for u_i: $u_i \sim \mathcal{U}(u_i|0, w_{c_i})$.
2. Conditionals for $\boldsymbol{\theta}_k^*$ and w_k : they are the same that in the truncated blocked Gibbs sampler of [IJ01] previously described.
3. Conditional for c_i: $\Pr(c_i = k) \propto \mathbf{1}(k : w_k > u_i)\ p(\mathbf{x}_i|\boldsymbol{\theta}_k^*)$. To sample from this probability mass function, one needs to know the exact number of components that are required at each iteration of the sampler. It is given by the smallest K such that

$$\sum_{k=1}^{K} w_k > 1 - u^*,$$

where $u^* = \min\{u_1, \ldots, u_n\}$.

In this section, we have outlined a marginal and two conditional methods for inference in Pitman-Yor mixture models. In the next section, the merits and limitations of each class of algorithms are summarized. This motivates the development of a new sampling approach which falls within the class of conditional approaches.

4 A new sampling method

By integrating mixture components out of the model, marginal algorithms make the allocation step very sequential since they need to condition on all previously allocated data. These incremental updates make marginal samplers not easily parallelizable. This feature is prejudicial when working with large data sets since the allocation step forms the most time consuming part of the algorithm. However, marginal samplers have the advantage that they deal with exchangeable prediction rules and exhibit most of the time better mixing properties.

The stick-breaking representation of the Pitman-Yor process allows to explicitly represent the weights in terms of independent Beta random variables. As a consequence, updating indicator variables in the allocation step is done without conditioning on the other indicators. This makes conditional algorithms using this representation easy to implement in a parallel computer. However this simplicity in the representation comes at a cost of slower mixing, since the stick-breaking prior has a weak preference for components to be sorted by decreasing mass. Consequently, additional moves in the MCMC algorithm are

necessary to improve mixing over clusters labels as suggested in [PISW06] and [Pap08].

In this context, we propose a conditional algorithm which is rather different from the others discussed so far. Our sampler is an attempt to combine the main advantages of marginal and conditional algorithms. The underlying idea is to integrate out the explicit order of clusters labels like in marginal methods hence collapsing the model to a lower dimensional space while keeping components weights as done in conditional approaches. To these aims, we propose to replace the standard posterior updating of the mixing measure based on the stick-breaking representation, with a posterior updating of Pitman-Yor processes under the class of Poisson-Kingman models introduced by [Pit03]. The next proposition summarizes this posterior characterization.

Proposition 2 *[Pit96b], Corollary 20*
Let $H \sim PY(d, \alpha, G_0)$ where G_0 is a diffuse probability measure s.t. $\mathbb{E}(H) = G_0$. Consider a sample $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n | H \sim H$. Let $\{\boldsymbol{\theta}_j^*\}_{j=1}^{k_n}$ denote the set of unique values of $\{\boldsymbol{\theta}_i\}_{i=1}^n$ and n_j the number of occurrences of $\boldsymbol{\theta}_j^*$ in the sample. Then the posterior of H can be expressed as follows

$$H|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n \stackrel{d}{=} \sum_{j=1}^{k_n} w_j \delta_{\boldsymbol{\theta}_j^*} + r_{k_n} H_{k_n}, \tag{10}$$

where

$$\begin{aligned} (w_1, \ldots, w_{k_n}, r_{k_n}) &\sim Dir(n_1 - d, \ldots, n_{k_n} - d, \alpha + dk_n) \\ H_{k_n} &\sim PY(d, \alpha + dk_n, G_0), \end{aligned}$$

and H_{k_n} independent of $(w_1, \ldots, w_{k_n}, r_{k_n})$, with $\mathbb{E}(H_{k_n}) = G_0$.

The posterior characterization (10) allows us to work on the space of equivalence classes of clusters $\boldsymbol{\theta}_j^*$ and, due to exchangeability, to integrate out the order of cluster labels as in the marginal samplers. Indeed, Pitman showed in [Pit96a] equivalence between exchangeability of the random partition generated by sampling from a discrete distribution and symmetry in the law characterizing the limiting frequencies of occupied components given the data. We can easily check that exchangeability is ensured in equation (10) since it sums to a symmetric Dirichlet distribution and an unconditional Pitman-Yor process (independent of the observed data). So, our sampler lives in the space of equivalence classes over clusters labels. These labels are then exchangeable and no mix over them is needed. This property has important consequences on the algorithm mixing. As opposed, in the conditional algorithms using the stick-breaking representation, exchangeability is lost when using the usual updating rule:

$$H(\cdot)|\boldsymbol{\theta}_1, \ldots \boldsymbol{\theta}_n = \sum_{k \in \mathbf{c}^*} w_k^* \delta_{\boldsymbol{\theta}_k^*}(\cdot) + \sum_{k \notin \mathbf{c}^*} w_k \delta_{Z_k}(\cdot), \tag{11}$$

where $\mathbf{c}^* = (c_1^*, \ldots, c_{k_n}^*)$ are the unique values of the classification variables $\mathbf{c} = (c_1, \ldots, c_n)$, the weights w_k^* follow a GEM distribution with updated parameters: $w_1^* = v_1^*, w_2^* = v_2^*(1 - v_1^*), \ldots, w_n^* = v_n^* \prod_{i=1}^{n-1}(1 - v_i^*)$ where $v_l^* \sim \text{Beta}(1 - d + n_l, \alpha + ld + \sum_{m=l+1}^{\infty} n_m)$, and for all $k \notin \mathbf{c}^*$, $Z_k \overset{iid}{\sim} G_0$.

This clearly illustrates that the posterior distribution of a random probability measure constructed via the stick-breaking representation depends on which explicit atoms labels observations are allocated to. This property is not necessary and has the impact of bothering the Gibbs sampler.

Given the posterior characterization (10), we are now in position to set up a new Gibbs sampling scheme for simulating from the posterior of a Pitman-Yor mixture model.

4.1 Proposed variants

For simulating H_{k_n}, the continuous part of the posterior given in equation (10), we propose two variants. The first makes use of a thresholded version of the "slice efficient dependent" of [KGW11]. The second is based on a truncation as originally suggested in [IJ01].

4.1.1 Exchangeable Thresholded Slice Sampler

We first propose a slice sampler inspired from [Wal07] and [KGW11]. The main steps are now summarized.

We augment the state with additional slice variables $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ such that the joint density for any $(\mathbf{x}_i, u_i)$, given a collection $\mathbf{w}$ of random masses and component parameters $\boldsymbol{\Theta}^*$, is

$$f(\mathbf{x}_i, u_i) = \sum_{k=1}^{\infty} w_k \; p(\mathbf{x}_i|\boldsymbol{\theta}_k^*) \mathcal{U}(u_i|0, \xi_k), \tag{12}$$

where ξ_k is a dependent variable such that for all k,

$$\xi_k = \min(w_k, \zeta), \tag{13}$$

with $\zeta \in]0, 1]$ and is independent of w_k. Here, ζ is a threshold that we propose in order to improve mixing properties of the sampler compared to [Wal07] and [KGW11]. The threshold ζ can be a random or deterministic variable. Here, the role of ζ is to ensure that on average at each iteration, all occupied clusters and at least a non-occupied one are proposed by the algorithm. For example, a deterministic typical value of ζ that gives rise to good trade-off between mixing properties and computational burden is the mean weight of the first atom (in size-biased order of H_{k_n}) with no data allocated to, which can be expressed in the two-parameter case as

$$\zeta = \frac{(\alpha + d\,\mathbb{E}_{\alpha,d}(K_n))(1-d)}{(\alpha+n)(\alpha+1)},$$

where $\mathbb{E}_{\alpha,d}(K_n)$ is the expected value of K_n:

$$\mathbb{E}_{\alpha,d}(K_n) = \sum_{i=1}^{n} \frac{(\alpha+d)_{i-1\uparrow}}{(\alpha+1)_{i-1\uparrow}},$$

where $(x)_{a\uparrow} = \Gamma(x+a)/\Gamma(x)$ is the Pochhammer symbol.
In the case of the Dirichlet process ($d = 0$),

$$\mathbb{E}_{\alpha,d}(K_n) = \sum_{i=1}^{n} \frac{\alpha}{\alpha+i-1} = \alpha \log\left(1 + \frac{n}{\alpha}\right).$$

For $d \neq 0$, it can be easily checked that,

$$\mathbb{E}_{\alpha,d}(K_n) = \frac{\alpha}{d}\left(\frac{(\alpha+d)_{n\uparrow}}{(\alpha)_{n\uparrow}} - 1\right).$$

For sufficiently large n, this expectation can be fairly approximated using Stirling's formula

$$\mathbb{E}_{\alpha,d}(K_n) \approx \frac{\Gamma(\alpha+1)}{d\,\Gamma(\alpha+d)} n^d.$$

Coming back to the slice sampling formulation, using equation (13), we can rewrite equation (12) as follows:

$$f(\mathbf{x}_i, u_i) = \mathbf{1}(\zeta > u_i)\,\zeta^{-1} \sum_{w_k > \zeta} w_k\; p(\mathbf{x}_i|\boldsymbol{\theta}_k^*) + \sum_{w_k \leq \zeta} \mathbf{1}(w_k > u_i)\, p(\mathbf{x}_i|\boldsymbol{\theta}_k^*),$$

where both sums are finite since $\#\{j : w_j > \varepsilon\} < \infty$, for all $\varepsilon > 0$. The use of $\mathbf{u}$ allows to sample a finite number K^* of weights and locations for H_{k_n}.

Let us here denote $\mathbf{w} = (w_1, w_2, \ldots, w_{k_n}, \mathbf{w}_{k_n})$ where $w_1, w_2, \ldots, w_{k_n}$ are the k_n Dirichlet random weights in the posterior characterization (10), and $\mathbf{w}_{k_n}$ is a collection of random variables distributed according to a two-parameter GEM$(d, \alpha + dK_n)$ distribution; these are the stick-breaking random weights of H_{k_n}. The Gibbs sampler allows to generate variables from the joint posterior of $(\boldsymbol{\Theta}^*, \mathbf{c}, \mathbf{w}, \mathbf{u}|\mathbf{X})$, by sampling iteratively from each full conditional. As in [KGW11], we jointly sample $\mathbf{w}, \mathbf{u}|\mathbf{c}$. The full conditional distributions involved in the steps of the sampler are then:

- $p(c|\boldsymbol{\theta}^*, w, u)$,
- $p(\boldsymbol{\theta}^*|c, w, u)$,
- $p(w, u|c, \boldsymbol{\theta}^*) = p(u|w, c, \boldsymbol{\theta}^*)\, p(w|c, \boldsymbol{\theta}^*)$.

We now provide a way of simulating from each conditional.

1. Conditional for $(\mathbf{w}, \mathbf{u})$:
   We jointly sample $\mathbf{w}, \mathbf{u}|\mathbf{c}$ in three steps by first sampling $w_1, w_2, \ldots, w_{k_n}|\mathbf{c}$, then $\mathbf{u}|w_1, w_2, \ldots, w_{k_n}, \mathbf{c}$, and finally $\mathbf{w}_{K_n}|\mathbf{u}$. The mains steps are now given.

   - Sample w_k for $k \leq k_n$:

   $$w_1, \ldots, w_{K_n}, r_{k_n}|\mathbf{c} \sim \mathrm{Dir}(n_1 - d, \ldots, n_{k_n} - d,\, \alpha + k_n\, d).$$

   - Sample $u_i|w_1, w_2, \ldots, w_{k_n}, \mathbf{c}$:

   $$u_i|w_1, w_2, \ldots, w_{k_n}, \mathbf{c} \overset{\text{ind.}}{\sim} \mathcal{U}(u_i|0, \min(w_{c_i}, \zeta)).$$

   Set $u^* = \min\{u_1, \ldots, u_n\}$.

   - Sample w_k for $k > k_n$. While $r_{k-1} > u^*$,

$$
\begin{aligned}
v_k &\sim \text{Beta}\left(1-d, \alpha + k\,d\right), \\
w_k &= v_k\, r_{k-1}, \\
r_k &= r_{k-1}\left(1-v_k\right).
\end{aligned}
$$

Set $K^* = \min\left(\{k : r_k < u^*\}\right)$.

Clearly, $w_k < u^*$ for all $k > K^*$, that is why we only have to sample a finite set of w_{K^*}.

Note that, at each iteration, non-empty clusters are re-labeled according to their *order of appearance* in the sampling. We operate in the space of *equivalence classes over non-empty clusters labels* which are thus *exchangeable*. The stick-breaking prior only concerns empty clusters for the given iteration of the Gibbs sampler. As pointed out, this encourages good mixing over clusters.

2. Conditional for $\mathbf{c}$:
   As underlined, sampling of classification variables requires the computation of a normalizing constant which becomes feasible using auxiliary variables since the choice of c_i is from a finite set:

$$
c_i|\mathbf{w}, \mathbf{u}, \mathbf{\Theta}^*, \mathbf{X} \overset{\text{ind}}{\sim} \sum_{k=1}^{K^*} w_{k,i}\, \delta_k\left(\cdot\right),
$$

   where $w_{k,i} \propto \mathbf{1}\left(w_k > u_i\right) \max\left(w_k, \zeta\right) p\left(\mathbf{x}_i|\boldsymbol{\theta}_k^*\right)$, and $\sum_{j=1}^{K^*} w_{k,i} = 1$.

   Note also that, in order to speed up computations, it is convenient to sort weights w_k, $k > k_n$ in decreasing order. By this, we can avoid tests for all $k > \kappa$ as soon as $w_\kappa < u_i$.

3. Conditional for $\mathbf{\Theta}^*$:

   - Updating parameters for non-empty components from the density proportional to:

$$
G_0(\mathrm{d}\boldsymbol{\theta}_k^*) \prod_{i:c_i=k} p(\mathbf{x}_i|\boldsymbol{\theta}_k^*) \text{ for all } k \leq k_n.
$$

   - Sampling parameters for unallocated components from their priors:

$$
\boldsymbol{\theta}_k^* \overset{\text{iid}}{\sim} G_0, \text{ for } k_n < k \leq K^*.
$$

The blocked Gibbs sampler structure allows easy implementation of the algorithm on a parallel computer.

4.1.2 Exchangeable Truncated Gibbs Sampler

The second variant of the algorithm we propose is an alternative of the first one. It is still based on the posterior given in equation (10). But instead of using the slice sampling strategy to sample the continuous part H_{k_n}, we resort to an approximation by taking a fixed level L. This truncation eliminates the need

of auxiliary variables. This scheme was suggested in [IJ01]. We approximate equation (10) by

$$\sum_{j=1}^{k_n} w_j \delta_{\boldsymbol{\theta}_j^*} + r_{k_n} H_{k_n}^*,$$

where $H_{k_n}^*$ is an approximation of H_{k_n}, i.e a truncation of H_{k_n} at level L. The total number of represented components is then $K^* = k_n + L$. The main steps are now given.

- Sample classification variables:

$$(c_i|\mathbf{w}, \mathbf{u}, \boldsymbol{\Theta}^*, \mathbf{X}) \overset{\text{ind}}{\sim} \sum_{k=1}^{K^\star} w_{k,i}\, \delta_k(\cdot),$$

  where

$$w_{k,i} \propto w_k\; p(\mathbf{x}_i|\boldsymbol{\theta}_k^*) \quad \text{and} \quad \sum_{k=1}^{K^*} w_{k,i} = 1.$$

- Sample w_k for $k \leq k_n$:

$$(w_1, w_2, \ldots, w_{k_n}, r_{k_n}|\mathbf{c}) \sim \text{Dir}(n_1 - d, n_2 - d, \ldots, n_{k_n} - d,\, \alpha + k_n\, d).$$

- Sample w_k for $k_n < k \leq K^*$:

$$\begin{aligned} v_k &\sim \text{Beta}(1 - d, \alpha + k\, d), \\ w_k &= v_k\, r_{k-1}, \\ r_k &= r_{k-1}(1 - v_k). \end{aligned}$$

  Set $w_{K^*} = r_{K^*-1}$ such that $v_{K^*} = 1$.

- Sample components parameters using
  - the density proportional to

$$G_0(\mathrm{d}\boldsymbol{\theta}_k^*) \prod_{i:c_i=k} p(\mathbf{x}_i|\boldsymbol{\theta}_k)$$

    for non-empty components (i.e. $k \leq k_n$),
  - the priors for unallocated components:

$$\boldsymbol{\theta}_k^* \overset{\text{iid}}{\sim} G_0, \text{ for } k_n < k \leq K^*.$$

5 Comparisons of algorithms

In this section, we carry out a comparative study that involves a variety of data sets, both real and simulated. We evaluate the performance of the samplers described in the previous sections and our new sampling method. We thus compare these following algorithms:

- algorithm 8 of [Nea00] ("Algo. 8"),

- the slice efficient of [KGW11] ("Slice efficient"),
- the truncated blocked Gibbs sampler of [IJ01] ("Trunc."),
- the two variants of the proposed sampling scheme based on an exchangeable model ("Slice exch. thres." and "Trunc. exch.").

We also investigate the gain in the mixing performances of the algorithms due to the exchangeability property of the model on one hand, and to the proposed threshold on the other hand. For this reason, we implement in addition our slice sampler using the exchangeable model but without the threshold ("Slice exch. without thres.") and the slice efficient of [KGW11] which uses an non-exchangeable model with the introduction of the threshold ("Slice eff. thres."). Note that the "Slice efficient" is referred to as "Slice efficient dependent" in [KGW11], in contrast to their independent version which makes use of a deterministic slice function.

Data specification:

We tested the algorithms with $p(\cdot|\boldsymbol{\theta})$ being a normal kernel with parameters $\boldsymbol{\theta}^* = (\mu, \sigma^2)$ and G_0 a normal-inverse Gamma distribution i.e, $G_0(\mu, \sigma^{-2}) = \mathcal{N}(\mu|\eta, \kappa^2) \times \mathcal{G}(\sigma^{-2}|\gamma, \beta)$ where $\mathcal{G}(\cdot|\gamma, \beta)$ denotes the Gamma distribution with density proportional to $x^{\gamma-1}e^{-x/\beta}$.

For comparison purposes, we considered the same real and simulated data sets as in [KGW11].

1. The simulated data were generated from the following mixtures of Gaussians.
   - A bimodal mixture (bimod):
   $$0.5\mathcal{N}(-1, 0.5^2) + 0.5\mathcal{N}(1, 0.5^2).$$
   - An unimodal lepto-kurtic mixture (lepto):
   $$0.67\mathcal{N}(0, 1) + 0.33\mathcal{N}(0.3, 0.25^2).$$

   These simulated densities are shown in Fig.2.

   In order to gauge algorithms performance for small and large data sets, we generated $n = 100$, $n = 1,000$ and $n = 10,000$ draws from each of these two mixtures.
2. The real data are
   - Galaxy data, which are the velocities (in 10^3 km/s) of 82 distant galaxies diverging from our own. It is a popular data set in density estimation problems and is also used by [EW95], [GR01] for instance.
   - S&P: this consists of 2023 daily index returns. This data set is unimodal, asymmetric, and heavy-tailed.

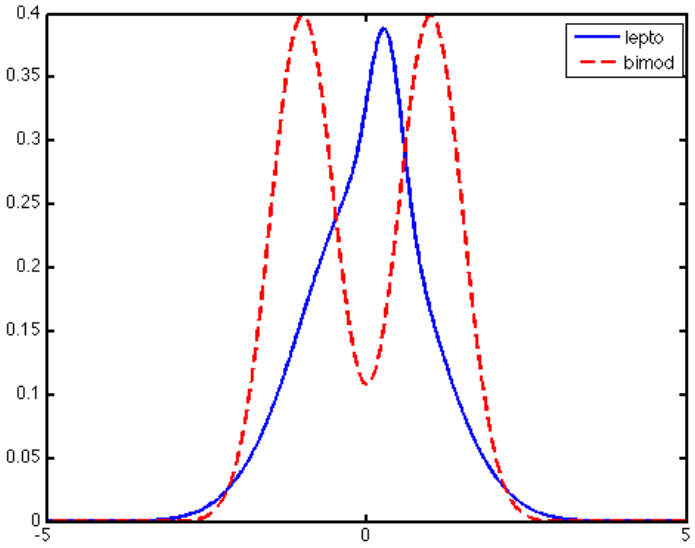

Figure 2: Bimodal (bimod) and unimodal lepto-kurtic(lepto) mixtures

Algorithms performance:

We monitored the convergence of two quantities: the deviance of the estimated density and the number of occupied clusters. The deviance is a global function of all parameters of the model and is defined as

$$D = -2\sum_{i=1}^{n} \log\left(\sum_{j} \frac{n_j}{n} p(\mathbf{x}_i|\boldsymbol{\theta}_j^*)\right),$$

where n_j is the size of cluster j.

The performance of competing samplers in their stationary regime was judged by looking at the integrated autocorrelation time (IAT) for each monitored quantity. IAT is defined in [Sok97] as,

$$\tau = 1 + 2\sum_{j=1}^{\infty} \rho_j,$$

where ρ_j is the sample autocorrelation at lag j. This quantity is an indicator of mixing behaviour of algorithms and measures *effectiveness* of MCMC samples. As such, it has also been used by other authors to compare MCMC methods (for example [Nea00], [GR01], [PR07], [KGW11]). IAT controls the statistical error in Monte Carlo measurements. In fact, the correlated samples generated by a Markov chain at equilibrium cause a variance that is 2τ larger than in independent sampling [Sok97]. If we denote by τ_j the integrated autocorrelation time produced by algorithm j for a given quantity, then $\tau_1/\tau_2 = k > 1$ means

that algorithm 1 requires k more iterations than algorithm 2 to produce the same Monte Carlo error [PR07]. So, when comparing two alternative Monte Carlo algorithms for the same problem, the most efficient is the one that produces the smaller IAT since it provides better estimates.

However, the calculation of IAT is difficult in practice. Following [Sok97], an estimator of τ can be obtained by summing the estimated autocorrelations up to a fixed lag L:

$$\hat{\tau} = 1 + 2\sum_{j=1}^{L} \hat{\rho}_j.$$

The choice of the cut-off point L is left to the user.

One can also estimate the standard error of $\hat{\tau}$ using this formula from [Sok97]:

$$\text{std}(\hat{\tau}) \approx \sqrt{\frac{2(2L+1)}{M}\tau^2}$$

where M is the Monte-Carlo size.

Algorithms parametrization:

At first, we set the discount parameter d of the PYM to zero in order to reduce it to a DPM. The strength parameter of the PYM, that is now the precision parameter of the DPM, was respectively set to $\alpha = \{1, 5\}$. Secondly, we investigated the behaviour of the competing algorithms in a power-law case (Pitman-Yor). The values $d = 0.3$ and $\alpha = 1$ were chosen for the PYM. The expected number of a priori components are then much larger than in the DPM case. We report in Tables 1-3 the expected number of a priori components, for each data set length and each parametrization.

Data	$\mathbb{E}(K_n)$
Galaxy ($n = 82$)	4.4
Lepto/bimod ($n = 100$)	4.6
Lepto/bimod ($n = 1,000$)	6.9
S&P 500 ($n = 2023$)	7.6
Lepto/bimod ($n = 10,000$)	9.2

Table 1: $\mathbb{E}(K_n)$ for each data set length in a DP($\alpha = 1$)

Data	$\mathbb{E}(K_n)$
Galaxy ($n = 82$)	14.3
Lepto/bimod ($n = 100$)	15.2
Lepto/bimod ($n = 1,000$)	26.5
S&P 500 ($n = 2023$)	30
Lepto/bimod ($n = 10,000$)	38

Table 2: $\mathbb{E}(K_n)$ for each data set length in a DP($\alpha = 5$)

Data	$\mathbb{E}(K_n)$
Galaxy ($n = 82$)	10.63
Lepto/bimod ($n = 100$)	11.48
Lepto/bimod ($n = 1,000$)	25.5
S&P 500 ($n = 2023$)	36.4
Lepto/bimod ($n = 10,000$)	58.9

Table 3: $\mathbb{E}(K_n)$ for each data set length in a PYP($\alpha = 1, d = 0.3$)

The hyperparameters have been fixed in a data-driven way according to [GR01] and set as follows: if R is the range of the data we take $\eta = R/2$ (mid-range), $\kappa^2 = 1/R^2$, $\gamma = 2$ and $\beta = 0.02R^2$.

The blocked Gibbs sampler of [IJ01] was truncated at level $N = 3\alpha\log(n)$, where n is the data size. This induces a truncation error that stands for the L_1 distance between the marginal density of the data under the truncated model and the marginal density under the full model, (see [IJ01]). The corresponding truncation errors for the different data sets are reported in the following table.

Data	ϵ
Galaxy ($n = 82$)	7.4139e-04
Lepto/bimod ($n = 100$)	9.0413e-04
Lepto/bimod ($n = 1,000$)	8.2446e-06
S&P 500 ($n = 2023$)	2.2572e-06
Lepto/bimod ($n = 10,000$)	7.5181e-08

We also truncated the second variant of our sampler at level $L = 2\alpha\log(n)$. The algorithm 8 of [Nea00] was tested with $m = 2$ auxiliary components.

We followed the instructions of [Sok97] who recommends running the samplers for a sufficient number of iterations. For each of the data sets, we ran $2,000,000$ iterations for each algorithm and discarded the first $200,000$ for the burn-in period. We believe that these numbers are sufficient to obtain reliable results.

Results and comments:

We report in Tables 4-9 the results of our comparisons for each set of data in the DPM case with $\alpha = 1$. The other results are postponed in the appendix. Each table contains respectively, for each algorithm, the estimated IAT for the mean number of clusters and for the deviance, the estimated mean number of clusters and the estimated deviance. Estimated IAT are obtained by integrating autocorrelation values for each monitored quantity up to a fixed lag (L_D for deviance and L_C for the mean number of clusters). The estimates of standard errors are put inside parentheses.

For visual comparison purposes, the autocorrelation curves are displayed in Figures 4 and 5 for Galaxy data. We also show in Figure 3 the histogram of the data and the density estimates when using each algorithm. By looking at the curves of the estimated densities, the values of the estimated deviances and the mean number of clusters, we made sure that all algorithms perform the estimation correctly and then they can be assessed through their mixing performance.

In the overall experiments, it turns out that:

- As expected, algorithm 8 performs better than all conditional algorithms since it works in an unidentifiable allocation structure. Furthermore, integrating out the mixture components speeds up the convergence since the dimensionality of the space is drastically reduced. One can refer to [PR07] and [PISW06] for more details about why conditional approaches are outperformed by marginals.

- On the other hand, the two variants of our method are superior to all other competitors in conditional algorithms using the stick-breaking representation, thanks to exchangeability in the model and the introduction of the threshold we propose. The "Slice efficient" gives the worst performance.

We believe that the poor-mixing due to non-exchangeability in the posterior stick-breaking representation is emphasized by the lack of the weights in slice samplers. This could often hinder the Gibbs sampler in the allocation step, for changing an observation from a component associated with a few observations to a component associated with many. Introducing our threshold would facilitate this change. To validate this conjecture, we experimented the effect of the threshold in the "Slice efficient". This algorithm is referred as "Slice eff. thres." Furthermore, the threshold makes little difference between the thresholded slice efficient ("Slice eff. thres.) and the truncated blocked Gibbs sampler (Trunc.). This later considers the weights of the mixture components when updating classification variables.

On the flip side, removing the threshold in our sampler ("Slice exch. without thres.") increases the IAT. Overall, it was observed on all data sets that the threshold causes a rapid decrease of autocorrelation curves in the first lags. However, it slightly increases the computation time per iteration. We underline that all algorithms have been implemented without any parallelization. All of them, excluding "Algo. 8", may be easily parallelized.

We now turn our attention to the benefits we reap thanks to the exchangeability property of the model. This is notable in differences between "Slice exch. thres." and "Slice eff. thres." and in differences between "Trunc. exch." and "Trunc.". We also notice on the curves that the autocorrelations obtained by "Slice exch. without thres." decrease and reach zero faster than in algorithms using non-exchangeable models ("Trunc.", "Slice eff. thres" and "Slice efficient"). This behaviour was observed on all data sets.

It is worth noting that the two variants of our algorithms and algorithm 8 of [Nea00] were stable in all experiments: for various simulations, we always obtained the same results in each data set and in each size of data. On the contrary, the algorithms using non-exchangeable models ([IJ01] and [KGW11]) did not always give the same results. We also observed erratic convergence behaviour of the Gibbs sampler in these two algorithms, particularly for large data sets (for example lepto with $n = 10,000$).

Results for $d = 0$ and $\alpha = 1$

In the following tables, n stands for the data set length, L_D and L_C are respectively the number of autocorrelation lags for deviance and for the clusters number. Values inside parentheses correspond to standard deviations of estimates.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	14.48(0.37)	2.88(0.05)	3.986(0.93)	1561.14(21.61)
Trunc. exch.	14.42(0.37)	2.94(0.05)	3.989(0.93)	1561.16(21.69)
SE without thres.	35.52(0.92)	4.77(0.09)	3.989(0.93)	1561.15(21.61)
Truncated	38.65(1.00)	3.63(0.07)	3.996(0.94)	1561.15(21.66)
Slice efficient	60.65(1.57)	5.28(0.10)	3.991(0.93)	1561.15(21.62)
Slice eff. thres.	37.82(0.98)	3.61(0.07)	3.986(0.93)	1561.08(22.17)
Algo 8 ($m = 2$)	8.25(0.21)	2.57(0.05)	3.987(0.93)	1561.16(21.62)

Table 4: Galaxy data $n = 82$, $L_D = 150$, $L_M = 300$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	28.76(0.74)	5.85(0.11)	3.801(1.66)	287.59(8.46)
Trunc. exch.	28.51(0.74)	6.00(0.11)	3.808(1.67)	287.58(8.48)
SE without thres.	70.56(1.82)	9.54(0.17)	3.799(1.67)	287.58(8.43)
Truncated	54.38(1.40)	5.89(0.11)	3.789(1.66)	287.58(8.42)
Slice efficient	99.92(2.58)	8.76(0.16)	3.784(1.65)	287.58(8.42)
Slice eff. thres.	55.41(1.43)	5.65(0.10)	3.794(1.66)	287.61(8.58)
Algo 8 ($m = 2$)	15.59(0.40)	5.20(0.09)	3.794(1.66)	287.59(8.52)

Table 5: Bimod data $n = 100$, $L_D = 150$, $L_M = 300$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	93.28(3.93)	3.65(0.07)	3.806(1.73)	2735.14(8.66)
Trunc. exch.	91.20(3.85)	3.69(0.07)	3.795(1.72)	2735.14(8.66)
SE without thres.	228.64(9.64)	5.44(0.10)	3.809(1.73)	2735.15(8.67)
Truncated	156.27(6.60)	3.71(0.07)	3.777(1.71)	2735.13(8.62)
Slice efficient	257.25(10.85)	5.13(0.09)	3.766(1.68)	2735.12(8.61)
Slice eff. thres.	150.13(6.33)	3.81(0.07)	3.798(1.71)	2735.15(8.72)
Algo 8 ($m = 2$)	47.25(1.99)	3.06(0.06)	3.798(1.72)	2735.14(8.65)

Table 6: Bimod data $n = 1000$, $L_D = 150$, $L_M = 800$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	25.61(0.85)	13.53(0.29)	3.991(1.64)	239.74(11.38)
Trunc. exch.	24.78(0.83)	13.46(0.28)	3.983(1.63)	239.75(11.31)
SE without thres.	90.56(3.02)	42.74(0.90)	3.991(1.63)	239.73(11.26)
Truncated	41.22(1.37)	17.03(0.36)	4.001(1.64)	239.72(11.31)
Slice efficient	120.71(4.03)	46.28(0.98)	3.979(1.64)	239.77(11.29)
Slice eff. thres.	44.73(1.49)	16.98(0.36)	3.989(1.64)	239.77(11.82)
Algo 8 ($m = 2$)	14.79(0.49)	9.83(0.28)	3.994(1.63)	239.72(11.36)

Table 7: Lepto data $n = 100$, $L_D = 200$, $L_M = 500$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	235.49(9.93)	13.17(0.24)	4.006(2.05)	2400.51(18.68)
Trunc. exch.	237.70(10.02)	13.66(0.25)	4.022(2.08)	2400.48(18.72)
SE without thres.	462.09(19.49)	18.75(0.34)	3.973(1.99)	2400.53(18.73)
Truncated	294.24(12.41)	12.67(0.23)	3.958(2.01)	2400.47(18.49)
Slice efficient	472.95(19.95)	16.91(0.31)	3.864(1.92)	2400.45(18.26)
Slice eff. thres.	302.50(12.76)	13.80(0.25)	3.978(2.07)	2400.53(18.93)
Algo 8 ($m = 2$)	148.81(6.28)	11.55(0.21)	4.018(2.07)	2400.48(18.69)

Table 8: Lepto data $n = 1000$, $L_D = 150$, $L_M = 800$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	22.58(0.75)	145.07(3.06)	4.977(0.82)	14990.47(57.56)
Trunc. exch.	21.59(0.72)	148.69(3.14)	4.978(0.82)	14990.50(59.09)
SE without thres.	93.93(3.13)	194.26(4.10)	4.976(0.82)	14990.35(57.80)
Truncated	32.75(1.09)	148.66(3.14)	4.975(0.81)	14990.94(59.44)
Slice efficient	105.92(3.53)	204.63(4.32)	4.965(0.81)	14991.21(61.14)
Slice eff. thres.	34.87(1.16)	145.46(3.07)	4.969(0.82)	14990.56(60.53)
Algo 8 ($m = 2$)	13.55(0.45)	106.28(2.24)	4.980(0.82)	14990.38(59.09)

Table 9: S&P 500 $n = 2023$, $L_D = 200$, $L_M = 500$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	244.64(11.53)	5.79(0.12)	3.77(1.74)	27235.46(9.09)
Trunc. exch.	247.02(11.65)	5.57(0.12)	3.77(1.73)	27235.45(9.04)
SE without thres.	433.42(20.44)	11.82(0.25)	3.75(1.73)	27235.44(9.01)
Truncated	286.67(13.52)	5.46(0.11)	3.73(1.74)	27235.44(8.98)
Slice efficient	456.95(21.55)	12.20(0.26)	3.76(1.76)	27235.44(9.00)
Slice eff. thres.	258.92(12.21)	6.09(0.13)	3.75(1.72)	27235.46(9.09)
Algo 8 ($m = 2$)	180.58(8.51)	4.76(0.10)	3.78(1.76)	27235.46(9.05)

Table 10: Bimod data $n = 10,000$, $L_D = 200$, $L_M = 1000$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	212.65(10.03)	11.74(0.25)	3.74(1.77)	23517.95(12.52)
Trunc. exch.	179.46(8.46)	10.96(0.23)	3.74(1.74)	23517.94(12.43)
SE without thres.	502.18(23.68)	19.68(0.42)	3.82(1.82)	23518.00(12.64)
Truncated	186.83(8.81)	17.80(0.38)	4.73(1.75)	23518.57(13.92)
Slice efficient	444.243(20.95)	17.60(0.37)	3.68(1.70)	23517.90(12.35)
Slice eff. thres.	203.24(9.58)	12.03(0.25)	3.73(1.76)	23517.99(12.59)
Algo 8 ($m = 2$)	142.67(6.73)	10.47(0.22)	3.74(1.77)	23517.95(12.47)

Table 11: Lepto data $n = 10,000$, $L_D = 200$, $L_M = 1000$.

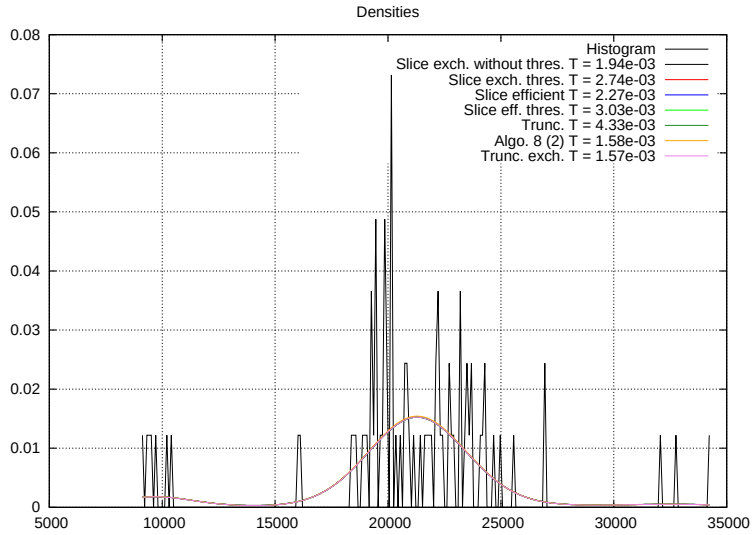

Figure 3: Histogram of data and estimated densities (Galaxy data).

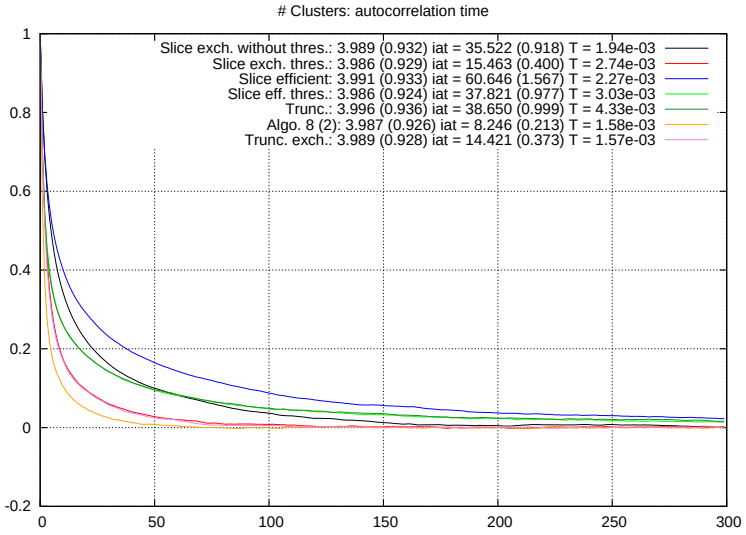

Figure 4: Autocorrelation curves used to estimate the IAT for the number of clusters (Galaxy data).

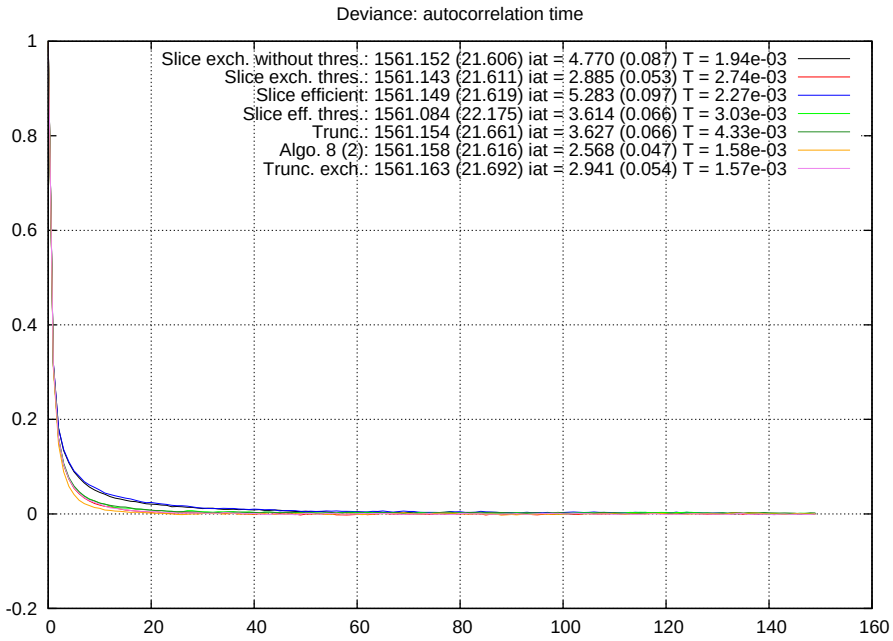

Figure 5: Autocorrelation curves used to estimate the IAT for the deviance (Galaxy data).

6 Conclusion and discussion

When models become more and more complex, due to an increase in dimension, the poor mixing of a MCMC algorithm can be inhibiting. Therefore, it seems important to develop samplers that allow to improve mixing while experimenting strategies to reduce the computational cost. The present paper has aimed at providing a simple, efficient and easy to use Gibbs sampler for posterior simulation under Pitman-Yor and Dirichlet mixture models that satisfies the constraint of efficient parallelization ability while maintaining mixing properties closed to Pólya urn approaches. We have attempted to combine blocking properties of conditional approaches which retain the random distribution in the sampling, and exchangeability of the model which is maintained in Pólya urn based algorithms. The proposed approach combines an update but unused formula from [Pit96b] with the slice sampling sampling strategy and a tricky threshold. Our comparative study on both real and simulated data sets support our belief that the two novel variants of our conditional Gibbs sampler have the potential to be a useful addition to the menu of samplers for DP and PYP.

A difference between the two proposed variants is that for the truncated version ("Trunc. exch."), the fixed length of approximation has to be decided before effective sampling. This is most of the time not a crux for Dirichlet processes, but for the two-parameter case the fixed approximation may give rise to biased estimates for moderate truncation lengths. For large lengths, the computational burden is emphasized especially for large data sets. The exchangeable thresholded slice version ("Slice exch. thres.") achieves adaptive truncation at each iteration and maintain nice trade-off between IAT and time cost. This latter variant gives then rise to convenient trade-off between IAT and computation time while avoiding any hard truncation.

On one hand, as mentionned our samplers are applicable to Pitman-Yor and Dirichlet mixture models. On the other hand, since the "Slice efficient" of [KGW11] has been developed for more general stick-breaking priors and that the introduction of the proposed threshold improves its mixing property, one can consider such a combination when working with mixtures based on general stick-breaking processes other than Dirichlet and Pitman-Yor processes. In this case, an interesting perspective could be to introduce also mixing moves over clusters labels as suggested in [PISW06] and [PR07]. As mentioned, the ordering of clusters labels matters in the stick-breaking representation. A step of labels permutation could result in a better mixing chain.

In our experimental study, it appeared that particularly for the two-parameter class, standard conditional algorithms may present unexpected biased results. This drawback is reinforced for large data sets. On the other hand, Pólya urn based algorithms and our proposed sampling schemes exhibit stable behaviour in all situations.

Appendix

In this appendix, we show the rest of the results from our comparative study, excepted results for lepto and bimod data with $n = 10,000$.

A-Results for $d = 0$ and $\alpha = 5$

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	10.73(0.28)	2.80(0.05)	7.082(3.32)	1563.10(23.55)
Trunc. exch.	10.11(0.26)	2.81(0.05)	7.084(3.31)	1563.10(23.54)
SE without thres.	26.32(0.68)	4.13(0.07)	7.085(3.32)	1563.10(23.53)
Truncated	19.51(0.50)	3.45(0.06)	7.079(3.32)	1563.10(23.57)
Slice efficient	38.75(1.00)	4.96(0.09)	7.085(3.31)	1563.11(23.59)
Slice eff. thres.	19.75(0.51)	3.32(0.06)	7.057(3.31)	1563.33(25.34)
Algo 8 ($m = 2$)	6.16(0.16)	2.35(0.04)	7.084(3.31)	1563.10(23.56)

Table 12: Galaxy data $n = 82$, $L_D = 150$, $L_M = 300$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	14.24(0.37)	2.35(0.04)	8.886(5.41)	283.18(8.98)
Trunc. exch.	13.74(0.35)	2.33(0.04)	8.880(5.38)	283.17(8.97)
SE without thres.	34.31(0.89)	3.30(0.06)	8.884(5.40)	283.17(8.95)
Truncated	23.22(0.60)	2.67(0.05)	8.883(5.41)	283.18(9.00)
Slice efficient	45.67(1.18)	3.58(0.06)	8.891(5.38)	283.18(8.97)
Slice eff. thres.	23.60(0.61)	2.38(0.04)	8.877(5.39)	283.56(9.98)
Algo 8 ($m = 2$)	8.56(0.22)	2.01(0.04)	8.888(5.42)	283.18(8.98)

Table 13: Bimod data $n = 100$, $L_D = 150$, $L_M = 300$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	81.36(3.43)	6.90(0.13)	9.889(7.26)	2741.07(12.05)
Trunc. exch.	81.07(3.42)	6.81(0.12)	9.956(7.35)	2741.08(12.05)
SE without thres.	200.17(8.44)	12.37(0.23)	9.913(7.29)	2741.08(12.07)
Truncated	135.98(5.73)	7.35(0.13)	9.958(7.34)	2741.08(12.09)
Slice efficient	256.71(10.83)	12.85(0.23)	9.962(7.40)	2741.09(12.06)
Slice eff. thres.	130.61(5.51)	6.92(0.13)	9.867(7.30)	2741.43(12.70)
Algo 8 ($m = 2$)	42.85(1.81)	5.35(0.10)	9.928(7.36)	2741.08(12.03)

Table 14: Bimod data $n = 1000$, $L_D = 150$, $L_M = 800$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	11.12(0.37)	14.26(0.30)	9.004(4.74)	257.17(21.70)
Trunc. exch.	11.84(0.39)	14.34(0.30)	8.988(4.75)	257.19(21.89)
SE without thres.	27.06(0.90)	30.15(0.64)	8.999(4.74)	257.15(21.64)
Truncated	17.79(0.59)	15.96(0.34)	9.011(4.75)	257.16(21.69)
Slice efficient	37.12(1.24)	33.49(0.71)	9.009(4.74)	257.18(21.67)
Slice eff. thres.	17.47(0.58)	15.16(0.32)	9.002(4.76)	257.56(23.14)
Algo 8 ($m = 2$)	6.95(0.23)	11.43(0.24)	8.999(4.73)	257.18(21.66)

Table 15: Lepto data $n = 100$, $L_D = 200$, $L_M = 500$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	90.94(3.84)	15.81(0.29)	11.121(7.69)	2354.96(19.51)
Trunc. exch.	90.68(3.82)	15.72(0.29)	11.127(7.64)	2354.95(19.59)
SE without thres.	216.32(9.12)	25.05(0.46)	11.082(7.67)	2354.91(19.34)
Truncated	145.95(6.16)	17.40(0.32)	11.080(7.70)	2354.98(19.60)
Slice efficient	254.45(10.73)	27.28(0.50)	11.196(7.65)	2354.90(19.60)
Slice eff. thres.	133.89(5.65)	15.62(0.29)	11.148(7.65)	2355.27(20.19)
Algo 8 ($m = 2$)	50.75(2.14)	13.13(0.24)	11.098(7.69)	2354.94(19.48)

Table 16: Lepto data $n = 1000$, $L_D = 150$, $L_M = 800$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	17.09(0.57)	151.34(3.91)	7.476(2.65)	14989.81(53.22)
Trunc. exch.	18.02(0.60)	143.45(3.71)	7.484(2.65)	14989.68(51.86)
SE without thres.	56.57(1.89)	214.64(5.55)	7.473(2.64)	14989.66(52.30)
Truncated	21.73(0.72)	150.94(3.90)	7.454(2.64)	14990.39(54.49)
Slice efficient	66.67(2.22)	225.38(5.82)	7.481(2.65)	14990.43(54.78)
Slice eff. thres.	23.14(0.77)	156.84(4.05)	7.475(2.65)	14990.18(56.20)
Algo 8 ($m = 2$)	11.33(0.38)	97.28(2.51)	7.478(2.65)	14989.76(52.48)

Table 17: S&P 500 data $n = 2023$, $L_D = 300$, $L_M = 500$.

B- Results for $d = 0.3$ and $\alpha = 1$

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	10.56(0.27)	2.84(0.05)	4.867(2.13)	1561.67(21.84)
Trunc. exch.	9.81(0.25)	2.79(0.05)	4.716(1.77)	1561.61(21.93)
SE without thres.	27.22(0.70)	4.57(0.08)	4.868(2.13)	1561.66(21.83)
Truncated	29.20(0.75)	3.65(0.07)	4.932(1.97)	1561.73(21.94)
Slice efficient	44.65(1.15)	5.43(0.10)	4.872(2.13)	1561.66(21.82)
Slice eff. thres.	24.95(0.64)	3.72(0.07)	4.858(2.11)	1561.79(23.21)
Algo 8 ($m = 2$)	5.79(0.15)	2.37(0.04)	4.869(2.13)	1561.66(21.89)

Table 18: Galaxy data $n = 82$, $L_D = 150$, $L_M = 300$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	25.77(0.67)	7.75(0.14)	4.726(3.42)	267.96(9.97)
Trunc. exch.	26.47(0.68)	7.97(0.15)	4.650(3.06)	267.93(9.99)
SE without thres.	67.54(1.74)	14.84(0.27)	4.715(3.42)	267.96(9.92)
Truncated	71.53(1.85)	9.88(0.18)	5.067(3.93)	267.87(10.00)
Slice efficient	97.86(2.53)	16.35(0.30)	4.743(3.42)	267.95(10.05)
Slice eff. thres.	56.18(1.45)	9.74(0.18)	4.710(3.42)	268.18(10.50)
Algo 8 ($m = 2$)	14.27(0.37)	5.79(0.11)	4.720(3.40)	267.95(9.99)

Table 19: Bimod data $n = 100$, $L_D = 150$, $L_M = 300$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	42.74(1.80)	2.60(0.05)	4.427(3.21)	2646.88(9.02)
Trunc. exch.	46.05(1.94)	2.54(0.05)	4.401(3.05)	2646.88(9.01)
SE without thres.	180.18(7.60)	5.86(0.11)	4.426(3.24)	2646.88(8.99)
Truncated	89.45(3.77)	2.82(0.05)	4.525(3.38)	2646.89(9.05)
Slice efficient	200.64(8.46)	6.23(0.11)	4.446(3.21)	2646.88(9.03)
Slice eff. thres.	77.30(3.26)	2.70(0.05)	4.409(3.18)	2647.10(9.45)
Algo 8 ($m = 2$)	22.80(0.96)	2.15(0.04)	4.425(3.18)	2646.88(8.99)

Table 20: Bimod data $n = 1000$, $L_D = 150$, $L_M = 800$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	47.85(1.60)	27.00(0.57)	3.719(3.70)	223.92(8.55)
Trunc. exch.	50.04(1.67)	26.91(0.57)	3.674(3.39)	223.86(8.61)
SE without thres.	188.12(6.27)	70.11(1.48)	3.709(3.69)	223.94(8.50)
Truncated	93.37(3.11)	35.96(0.76)	3.955(4.18)	223.78(8.75)
Slice efficient	224.68(7.49)	74.51(1.57)	3.696(3.68)	223.94(8.49)
Slice eff. thres.	85.39(2.85)	31.98(0.67)	3.732(3.73)	224.05(9.08)
Algo 8 ($m = 2$)	27.28(0.91)	17.83(0.38)	3.720(3.71)	223.92(8.55)

Table 21: Lepto data $n = 100$, $L_D = 200$, $L_M = 500$.

	IAT for # of clusters	*IAT for deviance*	*Estimated # of clusters*	*Estimated deviance*
Slice exch. thres	156.21(7.37)	13.56(0.29)	4.255(3.22)	2371.35(16.11)
Trunc. exch.	167.13(7.88)	13.08(0.28)	4.247(3.13)	2371.34(16.03)
SE without thres.	341.73(16.11)	18.15(0.38)	4.252(3.18)	2371.34(15.97)
Truncated	270.51(12.75)	17.04(0.36)	4.387(3.62)	2371.52(16.49)
Slice efficient	422.09(19.90)	20.47(0.43)	4.291(3.30)	2371.50(16.53)
Slice eff. thres.	217.35(10.25)	13.66(0.29)	4.226(3.19)	2371.54(16.60)
Algo 8 ($m = 2$)	96.90(4.57)	10.90(0.23)	4.242(3.22)	2371.34(15.98)

Table 22: Lepto data $n = 1000$, $L_D = 200$, $L_M = 1000$.

References

[BM73] D. Blackwell and J. B. MacQueen. Ferguson distributions via Pólya urn schemes. *Ann. Statist.*, 1:353–355, 1973.

[BM96] C. A. Bush and S. N. MacEachern. A semiparametric Bayesian model for randomised block designs. *Biometrika*, 83(2):275–285, 1996.

[CTM] F. Caron, Y.W. Teh, and T.B. Murphy. Bayesian nonparametric Plackett-Luce models for the analysis of preferences for college degree programmes. *Annals of Applied Statistics*, To appear.

[DBFL$^+$] P. De Blasi, S. Favaro, A. Lijoi, R. Mena, I. Prünster, and M. Ruggiero. Are Gibbs-type priors the most generalization of the Dirichlet process? *IEEE Transactions of Pattern Analysis and Machine Intelligence*.

[Esc94] M.D. Escobar. Estimating normal means with a Dirichlet process prior. *J. Am. Stat. Assoc.*, 89:268–277, 1994.

[EW95] M.D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *J. Am. Stat. Assoc.*, 90:577–588, 1995.

[Fer73] T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *Ann. Statist.*, 1:209–230, 1973.

[Fer83] T.S. Ferguson. Bayesian density estimation by mixtures of Normal distributions. *Recent advances in Statistics: papers in honor of Herman Chernoff on his sixtieth birthday*, pages 287–302, 1983.

[FK72] T.S. Ferguson and M.J. Klass. A representation of independent increment processes without Gaussian components. *The Annals of Mathematical Statistics*, 43(5):1634–1643, 1972.

[GR01] P. J. Green and S. Richardson. Modelling heterogeneity with and without the Dirichlet process. *Scandinavian Journal of Statistics*, 28:355–375, 2001.

[GW11] J. E. Griffin and S. G. Walker. Posterior simulation of Normalized Random Measure mixtures. *Journal of Computational and Graphical Statistics*, 20:241–259, 2011.

[IJ01] H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *J. Am. Stat. Assoc.*, 96:161–173, 2001.

[IJ03] H. Ishwaran and L. F. James. Some further developments for stick-breaking priors: Finite and infinite clustering and classification. *Sankhya Series A*, 65:577–592, 2003.

[KGW11] M. Kalli, J. E. Griffin, and S. G. Walker. Slice sampling mixture models. *Statistics and Computing*, 21(1):93–105, 2011.

[KH73] R. Korwar, , and M. Hollander. Contributions to the theory of Dirichlet processes. *The Annals of Probability*, pages 705–711, 1973.

[Lo84] A. Y. Lo. On a class of Bayesian Nonparametric estimates: I. density estimates. *The Annals of Statistics*, 12:351–357, 1984.

[MM98] S. N. MacEachern and P. Müller. Estimating mixture of Dirichlet process models. *J. Comput. Graph. Stat.*, 7:223–238, 1998.

[MM13] P. Müller and R. Mitra. Bayesian Nonparametric Inference–Why and How. *Bayesian Analysis*, 8(2):269–302, 2013.

[MQ04] P. Müller and F. Quintana. Nonparametric Bayesian data analysis. *Statistical Science*, 19(1):95–110, 2004.

[MT98] P. Muliere and L. Tardella. Approximating distributions of random functionals of Ferguson-Dirichlet priors. *Canadian Journal of Statistics*, 26(2):283–297, 1998.

[NBP09] L. E Nieto-Barajas and I. Prünster. A sensitivity analysis for Bayesian nonparametric density estimators. *Statistica Sinica*, 19:685–705, 2009.

[Nea91] R. M. Neal. Bayesian mixture modeling. In *Maximum entropy and Bayesian Methods: Proceedings of the 11th International Workshop on Maximum Entropy and Bayesian Methods of Statistical Analysis, Seattle*, pages 197–211, 1991.

[Nea00] R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[Pap08] O. Papaspiliopoulos. A note on posterior sampling from Dirichlet mixture models. *Preprint*, 2008.

[PISW06] I. R. Porteous, A. Ihler, P. Smyth, and M. Welling. Gibbs sampling for (coupled) infinite mixture models in the stick breaking representation. In *UAI*. AUAI Press, 2006.

[Pit96a] J. Pitman. Random discrete distributions invariant under size-biased permutation. *Adv. Appl. Prob*, 28:525–539, 1996.

[Pit96b] J. Pitman. Some developments of the Blackwell-MacQueen urn scheme. In T.S. Ferguson et al., editor, *Statistics, Probability and Game Theory; Papers in honor of David Blackwell*, volume 30 of *Lecture Notes-Monograph Series*, pages 245–267. Institute of Mathematical Statistics, 1996.

[Pit02] J. Pitman. Combinatorial stochastic processes. Technical Report 621, Dept. Statistics, U.C. Berkeley, 2002.

[Pit03] J. Pitman. Poisson-Kingman partitions. *Statistics and Science: A Festschrift for Terry Speed, IMS Lectures Notes Monograph*, 40:1–34, 2003.

[PR07] O. Papaspiliopoulos and G. O. Roberts. Retrospective Markov chain sampling Monte Carlo methods for Dirichlet process hierarchical models. *Biometrika*, 95:169–186, 2007.

[PY97] J. Pitman and M. Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Ann. Proba.*, 25:855–900, 1997.

[Sok97] A. D. Sokal. Monte carlo methods in statistical mechanics: Foundations and new algorithms. *NATO Adv. Sci. Inst. Ser. B Phys.*, 361:131–192, 1997.

[Teh06] Y. W. Teh. A hierarchical Bayesian language model based on Pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 985–992. Association for Computational Linguistics, 2006.

[Wal07] S. G. Walker. Sampling the Dirichlet mixture model with slices. *Comm. Statist.*, 36:45–54, 2007.

[WD98] S.G. Walker and P. Damien. Sampling methods for Bayesian nonparametric inference involving stochastic processes. *Practical Nonparametric and Semiparametric Bayesian Statistics*, 133:243–254, 1998.

[WME94] M. West, P. Müller, and M. D. Escobar. Hierarchical priors and mixture models, with application in regression and density estimation. *Aspects of Uncertainty*, pages 363–386, 1994.