A Simple and Efficient Method for Sampling Mixture Models based on Dirichlet and Pitman-Yor processes

Mame Diarra Fall, Eric Barat

A Simple and Efficient Method for Sampling Mixture Models based on Dirichlet and Pitman-Yor processes

Mame Diarra Fall[1*] and Éric Barat[2]

[1*]Institut Denis Poisson, UMR CNRS, Université d'Orléans, France.
[2]Université Paris Saclay, CEA, List, F91120, Palaiseau, France.

*Corresponding author(s). E-mail(s): diarra.fall@univ-orleans.fr;
Contributing authors: eric.barat@cea.fr;

Abstract

We introduce a simple and efficient sampling strategy for the Dirichlet Process Mixture model (DPM) and its two-parameter extension, the Poisson-Dirichlet process mixture model, also known as the Pitman-Yor process Mixture model (PYM). Inference in DPM and PYM is usually performed using Markov Chain Monte Carlo (MCMC) methods, specifically the Gibbs sampler. These sampling methods are usually divided into two classes: marginal and conditional algorithms. Each method has its own merits and limitations. The aim of this paper is to propose a simple and effective strategy that combines the main advantages of each class. Extensive experiments on simulated and real data highlight that the proposed sampler is relevant and performs much better than its competitors.

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

1 Introduction

Bayesian nonparametrics (BNP) have gained popularity in a wide range of applications in statistics and machine learning (density estimation, clustering, image segmentation and reconstruction, language modelling, etc.) ([1], [2], [3], [4]). The Dirichlet Process Mixture (DPM) model [5], [6] is by far the most popular BNP model. In recent years, models going beyond the DPM have been proposed in the literature ([7], [8], [9], [10]).

Let $\mathbf{X} = (X_1, \ldots, X_n)$ be an n-dimensional sample of observations defined on a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ and taking values in some separable metric space $\mathcal{X}$. Let $\mathcal{F}$ be the space of all probability distributions in $\mathcal{X}$. A BNP mixture model is a random distribution taking values in $\mathcal{F}$ and defined as follows,

$$f(x) = \int p(x|\boldsymbol{\theta}) \mathrm{d}H(\boldsymbol{\theta}), \tag{1}$$

where $\{p(\cdot|\boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta\}$ is a family of non-negative (possibly multivariate) kernels on $\mathcal{X}$ 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 mixing distribution H is a discrete random probability measure. In this paper, we will focus mainly on the following two cases: (1) $H \sim \mathrm{DP}(\alpha, G_0)$, that is H is a Dirichlet process (DP) with parameters $\alpha > 0$ and base distribution $G_0 \in \mathcal{F}$; and (2) $H \sim \mathrm{PY}(d, \alpha, G_0)$, a Pitman-Yor process with discount parameter $d \in [0, 1)$, strength parameter $\alpha > -d$ and base measure G_0. The DP is recovered as a special case of the PYP when $d = 0$. PYM is an interesting alternative to DPM, allowing greater modelling flexibility.

Alternatively, we can write the model (1) under the PYP prior in the following hierarchical form,

$$\begin{aligned} X_i|\boldsymbol{\theta}_i &\overset{ind}{\sim} p(x_i|\boldsymbol{\theta}_i), \quad i = 1, \ldots, n \\ \boldsymbol{\theta}_i|H &\overset{iid}{\sim} H \\ H|d, \alpha, G_0 &\sim \mathrm{PY}(d, \alpha, G_0). \end{aligned} \tag{2}$$

The PYP can be written using its *stick-breaking representation*,

$$H(\cdot) = \sum_{k=1}^{\infty} w_k \delta_{\boldsymbol{\theta}_k^*}(\cdot), \tag{3}$$

where $\delta_{\boldsymbol{\theta}^*}(\cdot)$ denotes the Dirac measure giving mass 1 at $\boldsymbol{\theta}^*$; the weights are constructed according to the so-called GEM distribution as follows: $w_1 = v_1$, $w_j = v_j \prod_{i=1}^{j-1}(1 - v_i)$, for all j, $v_j \overset{ind}{\sim} \mathrm{Beta}(1 - d, \alpha + jd)$. The locations $(\boldsymbol{\theta}_j^*)_{j \geq 1}$ are i.i.d. G_0, and independent of the weights. By exploiting the discrete nature of H, PYM 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, we are interested in the posterior distribution of the random density f. However, the latter has no closed form, and inference is necessarily simulation-based. Markov Chain Monte Carlo (MCMC) methods are the gold standard in BNP models. There are many MCMC sampling algorithms, which can be roughly divided into two categories: marginal and conditional methods. The difference between the two lies in the way they deal with the infinite-dimensional mixing measure H. In marginal methods, H is analytically marginalized out, whereas in conditional methods, it is represented explicitly.

Marginal methods can be in turn subdivided 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 of conditional posterior distributions are simplified and can be performed analytically ([11], [12], [13], [14] and [15]). In non-conjugate models, however, posteriors can not be easily calculated. The sampling scheme is more difficult and requires elaborate techniques ([16], [17] and [18]). The reader is referred to [19] for a more complete overview and discussions of these methods. Neal [19] 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, called "Algorithm 8", is based on an augmentation scheme and extends the model to include auxiliary components that temporarily exist.

An alternative to marginal methods are conditional algorithms that explicitly represent the mixing measure using, for instance, its stick-breaking representation ([20], [21]). The challenge in conditional approaches is to deal with the countably infinite representation of H in equation (3). In [20], the authors resort to an approximation and truncate the mixing measure to a deterministic value. An alternative that avoids hard truncations was proposed by [22] who provided an approximation of the Dirichlet process by means of a random truncation of its stick-breaking representation. The same was proposed for the Pitman-Yor process by [23]. The idea of random truncation but avoiding the introduction of truncation errors as in the previously cited papers was developed in [24] with a Metropolis-Hastings sampling scheme, and in [21] using the *slice sampling* strategy. The latter algorithm was improved by [25] and [26].

One of the advantages of marginal methods is their simplicity and the fact that the number of components to be updated at each iteration is finite and deterministic. However, by integrating mixture components outside the model, marginal algorithms make the allocation step highly sequential, since they must condition on all previously allocated data. These incremental updates mean that marginal samplers are not easily parallelizable. This is detrimental when working with large datasets, as the allocation step is the most time-consuming part of the algorithm. Another drawback of marginalizing over the mixing measure is that computing posterior conditionals require additional sampling steps ([20], [27], [28]). However, marginal samplers have the advantage of handling exchangeable prediction rules, and the most advanced algorithms in these methods have better mixing properties than conditional methods. In addition, random weights are collapsed by marginalization, leading to a crucial reduction in the dimension of the parameters space.

The stick-breaking representation of the Pitman-Yor process allows weights to be explicitly represented in terms of independent Beta random variables. This property makes conditional algorithms using this representation capable of updating blocks of parameters, and easy to parallelize to take advantage of recent parallel computing hardware architectures, which is particularly suited to large data sets. However this simplicity in the representation comes at a cost of slower mixing. Indeed, in this representation, weights are explicitly defined by the prior and components are represented with a size-biased ordering on 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 clusters labels

efficiently to avoid any clustering bias. Authors in [29] recommend the systematic use of two additional Metropolis-Hastings moves ("label-swap" and "label-permute") 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 [25]). 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 for the sampler, since cluster labels are irrelevant.

It is worth mentioning that there are hybrid samplers that cannot be classified as marginal or conditional ([30], [31]). We also point out that, instead of using the stick-breaking representation for the underlying mixing measure, some conditional samplers exploit other constructive representations. For instance, the use of the so-called Ferguson and Klass representation [32] of independent increment processes has been considered in the literature (see for example [10] and [33], and the references therein for some contributions in this direction). These approaches are interesting in that they allow to consider classes of priors that are, in general, broader than Pitman-Yor and Dirichlet processes. However, they become non-trivial to implement, even when applied to the DPM. Given the importance of DPM and PYM, the dominant priors in Bayesian nonparametrics, it seems important to devote attention to the development of alternative, simple and efficient algorithms. In particular, the mixing properties of MCMC samplers are proving to be a key point for high-dimensional applications with large data sets.

The purpose of this paper is to provide a simple and effective way to infer DPM and PYM. We propose a conditional sampling scheme that is formulated in the space of equivalence classes on clusters labels where clusters identities are irrelevant. To sample the infinite part, we propose two variants. Extensive simulations show that the proposed methods outperform all competing conditional sampling algorithms. Finally, we point out that since the preliminary ideas of this work were presented in a technical report, they have been successfully used and applied by other authors, for example in [34], [35], or they have inspired other authors [36].

The rest of the paper is structured as follows. In Section 2, we present the two variants of the proposed MCMC algorithm. We evaluate the performance of the algorithms through an extensive study on real and simulated data in Section 3. We conclude the paper in Section 4 with discussions and extensions for further work. Additional results are presented in the appendices.

2 Proposed sampling method

The proposed conditional algorithm is different from the other conditional algorithms discussed so far. Our sampler attempts to combine the main advantages of the marginal and conditional algorithms. The underlying idea is to integrate out the explicit order of clusters labels as in marginal methods, thereby collapsing the model to a lower-dimensional space while retaining component weights as in conditional approaches. We replace the standard posterior updating of the mixing measure based on the stick-breaking representation, with a posterior update of Pitman-Yor processes under

the class of Poisson-Kingman models introduced by [37]. The following proposition summarizes this posterior characterization.

Proposition 1. *[38], 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 $\{\boldsymbol{\theta}_j^*\}_{j=1}^{k_n}$ be 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,

$$H|\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n \stackrel{d}{=} \sum_{j=1}^{k_n} w_j \delta_{\boldsymbol{\theta}_j^*} + r_{k_n} H_{k_n}, \tag{4}$$

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 (4) allows us to work on the space of clusters equivalence classes and, due to exchangeability, to integrate out the order of cluster labels as in marginal samplers. Indeed, Pitman showed in [39] the equivalence between exchangeability of the random partition generated by sampling from a discrete distribution and the symmetry of the law characterizing the limiting frequencies of the occupied components given the data. We can easily check that exchangeability is ensured in equation (4), since it sums up a symmetrical Dirichlet distribution and an unconditional Pitman-Yor process (independent of the observed data). So our sampler lives in the space of equivalence classes on clusters labels. These labels are therefore exchangeable and no mix over them is necessary. This property has important consequences for algorithm mixing, as we shall see in the comparative study. As opposed, in conditional algorithms using the stick-breaking representation, exchangeability is lost when using the usual updating rule:

$$H(\cdot)|\boldsymbol{\theta}_1, \ldots \boldsymbol{\theta}_n = \sum_{k \in \mathbf{c}^*} w_k^* \delta_{\boldsymbol{\theta}_k^*}(\cdot) + \sum_{k \notin \mathbf{c}^*} w_k \delta_{Z_k}(\cdot), \tag{5}$$

where $\mathbf{c}^* = (c_1^*, \ldots, c_{k_n}^*)$ are the unique values of the classification variables $\mathbf{c} = (c_1, \ldots, c_n)$, where $c_i = k$ iff $\boldsymbol{\theta}_i = \boldsymbol{\theta}_k^*$. The weights w_k^* are constructed as follows: $w_1^* = v_1^*, w_2^* = v_2^*(1 - v_1^*), \ldots, w_n^* = v_n^* \prod_{i=1}^{n-1}(1 - v_i^*)$ where $v_l^* \sim \text{Beta}(1 - d + n_l, \alpha + ld + \sum_{m=l+1}^{\infty} n_m)$, and for all $k \notin \mathbf{c}^*$, $Z_k \stackrel{iid}{\sim} G_0$. This clearly illustrates that the posterior distribution of a random probability measure constructed via the stick-breaking representation depends on the explicit labels of the atoms to which the observations are assigned. This property is not necessary and has the effect of hindering the Gibbs sampler.

Using the posterior characterization (4), we propose a Gibbs sampling scheme to sample from the posterior of a Pitman-Yor mixture model and the Dirichlet process

mixture as a special case. To simulate H_{k_n}, the continuous part of the posterior given in (4), we propose two variants of the sampler. The first makes use of a thresholded version of the "slice efficient dependent" of [26]. The second is based on a truncation of the process, as originally suggested in [20].

2.1 Variant 1: Exchangeable Thresholded Slice Sampler

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

$$f(x_i, u_i) = \sum_{k=1}^{\infty} w_k \; p(x_i|\boldsymbol{\theta}_k^*) \mathcal{U}(u_i|0, \xi_k), \tag{6}$$

where ξ_k is a variable such that for all k,

$$\xi_k = \min(w_k, \zeta), \tag{7}$$

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

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

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

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

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 n sufficiently large, 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 (7), we can rewrite (6) as follows,

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

where both sums are finite since $\#\{j : w_j > \varepsilon\} < \infty$, for all $\varepsilon > 0$.

Let us now 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 (4), and $\mathbf{w}_{k_n}$ is a collection of random variables distributed according to the 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 iteratively sampling from each full conditional. As in [26], we jointly sample $\mathbf{w}, \mathbf{u}|\mathbf{c}$. The full conditional distributions involved in the sampler steps 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 to simulate 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 $r_{k-1} > u^*$,

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

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

   It is clear that $w_k < u^*$ for all $k > k^*$, so we only need to sample a finite set of w_{k^*}.

   Note that at each iteration, the non-empty clusters are relabelled according to their order of appearance in the sampling. We operate in the space of *equivalence*

classes on the labels of non-empty clusters which are therefore *exchangeable*. The stick-breaking prior only applies to empty clusters for the given iteration of the Gibbs sampler. As we have pointed out, this encourages good mixing over clusters.

2. Conditional for $\mathbf{c}$.
   The 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,

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

   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, to speed up computations, it is convenient to sort the weights w_k, $k > k_n$ in decreasing order. This avoids testing for all $k > \kappa$ as soon as $w_\kappa < u_i$.

3. Conditional for $\boldsymbol{\Theta}^*$.

   - 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 structure of the block Gibbs sampler makes it easy to implement the algorithm on a parallel computer.

2.2 Variant 2: Exchangeable Truncated Gibbs Sampler

The second variant of the proposed algorithm is an alternative to the first. It is still based on the posterior (4). 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 M. This truncation eliminates the need for auxiliary variables. The right-hand side of (4) is approximated 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 M. The total number of components represented is then $k^* = k_n + M$. The main steps are now given.

- Sample classification variables:

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

where

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

- Sample w_k for $k \leq k_n$:

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

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

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

  Set $w_{k^*} = r_{k^*-1}$ 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,

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

After presenting the two variants of the proposed algorithm, in the next section we compare it with a marginal method and conditional samplers on various data sets.

3 Algorithms comparisons

In this section, we carry out a comparative study on various data sets, both real and simulated. We compare the two variants of the proposed conditional sampler (named respectively "Slice exch. thres." and "Trunc. exch."), with two other conditional samplers: the efficient slice sampler proposed by [26] ("Slice efficient"), and the truncated blocked Gibbs sampler of [20] ("Truncated"). Note that "Slice efficient" is referred to as "Slice efficient dependent" in [26], unlike their independent version which uses a deterministic slice function. We will also compare ourselves with Algorithm 8 by Neal [19] ("Algo. 8"), known as the best algorithm for marginal methods. As the latter is a marginal algorithm, it is not included in the comparison between conditional methods discussed here. Its results are therefore provided for information, to give an idea of expected performance.

3.1 Data specification:

We tested the algorithms with $p(\cdot|\boldsymbol{\theta})$ being a univariate normal kernel with parameters $\boldsymbol{\theta}^* = (\mu, \sigma^2)$ and G_0 a normal-inverse gamma base measure 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 synthetic datasets as in [26].

1. Synthetic data were simulated 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):

   $$0.67\mathcal{N}(0, 1) + 0.33\mathcal{N}(0.3, 0.25^2).$$

   The corresponding densities are shown in Figure 1.

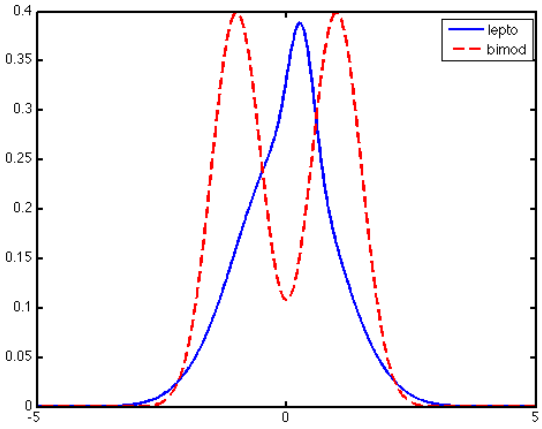


Fig. 1: Bimodal (bimod) and unimodal lepto-kurtic (lepto) mixtures.

In order to gauge the performance of the algorithms on small and large datasets, 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: these are the velocities (in 10^3 km/s) of 82 distant galaxies diverging from our own. This is a popular data set in density estimation problems.

- S&P 500: this consists of 2023 daily index returns. This data set is unimodal, asymmetric, and heavy-tailed.

3.2 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 model parameters 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 [40] 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 the mixing behaviour of the algorithms and measures the *efficiency* of an MCMC sampler. As such, it has also been used by many authors to compare MCMC methods (e.g., [19], [18], [24], [26]). IAT controls the statistical error in Monte Carlo measurements. In fact, correlated samples generated by a Markov chain at equilibrium cause a variance that is 2τ greater than in independent sampling [40]. 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 [24]. So, when comparing two alternative Monte Carlo algorithms for the same problem, the most efficient is the one that produces the smallest IAT, since it provides better estimates.

However, calculating IAT is difficult in practice. Following [40], an estimator of τ can be obtained by summing the estimated autocorrelations up to a fixed lag L,

$$\hat{\tau} = 1 + 2\sum_{j=1}^{L} \hat{\rho}_j. \tag{8}$$

One can also estimate the standard error of $\hat{\tau}$ using the following formula from [40],

$$\text{std}(\hat{\tau}) \approx \sqrt{\frac{2(2L+1)}{N}\tau^2} \tag{9}$$

where N is the Monte-Carlo size.

3.3 Algorithms parametrization

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, which is now the precision parameter of

the DPM, was respectively set to $\alpha = \{1, 5\}$. The prior expected number of components for each data set length and each parameterization are given in Tables 1-2.

Table 1: $\mathbb{E}(K_n)$ in DP($\alpha = 1$)

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 2: $\mathbb{E}(K_n)$ in DP($\alpha = 5$)

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

Secondly, we investigated the behaviour of competing algorithms in a PYM power-law case. In our experiments, we considered different parameter combinations. However, due to space constraints, we only present the results for $d = 0.3$ and $\alpha = 1$. Table 3 reports the prior expected numbers of clusters. Note that there are much higher than in the DPM case with $\alpha = 1$ (Table 1).

Table 3: $\mathbb{E}(K_n)$ in PYP($\alpha = 1, d = 0.3$)

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

The hyperparameters have been fixed in a data-driven way according to [18] and set as follows: if R is the data range, we take $\eta = R/2$ (mid-range), $\kappa^2 = 1/R^2$, $\gamma = 2$ and $\beta = 0.02R^2$.

The blocked Gibbs sampler of [20] ("Truncated") has been truncated to the level $K = 3\alpha\log(n)$, where n is the data size. We have also truncated the second variant of our sampler ("Trunc exch.") to the level $M = 2\alpha\log(n)$. Algorithm 8 of [19] was tested with $m = 2$ auxiliary components. We followed Sokal's instructions [40], which recommends running samplers for a sufficient number of iterations. For each data set, 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.

3.4 Results and comments

The following tables report the performance in terms of IAT achieved by

- the two variants of the proposed algorithm ("Slice exch. thres." and "Trunc. exch.")
- the truncated blocked Gibbs sampler of [20] ("Truncated").
- the efficient slice sampler of [26] ("Slice efficient").

IAT is calculated using formula (8), L will be respectively noted L_D for the IAT on deviance and L_C for that on the number of clusters. The estimated standard deviation (value in brackets in the tables) is calculated using (9). We recall that the smaller the IAT, the better the algorithm. The best performance is shown in bold and the second best in underlined (ranking is based on conditional algorithms). We also provide the mean number of clusters and the deviance estimated by each algorithm. This ensures that algorithms perform the estimation correctly, and that they can be assessed through their mixing performance. As mentioned, "Algorithm 8" is a marginal method and is not included in the comparison. Its performance is provided for information.

DPM case

We first look at the results for the DPM case, with $\alpha = 1$ (results with $\alpha = 5$ are provided in Appendix B), for real data (Galaxy and S&P 500) and simulated data (lepto and bimod with $n = 1,000$ and $n = 10,000$). Results are given in Tables 4-9. Simulation results for $n = 100$ are provided in Appendix A. This appendix also contains autocorrelation curves on Galaxy data, as well as the data histogram and estimated posterior density for each competing algorithm.

Table 4: Galaxy data $n = 82$, $L_D = 150$, $L_M = 300$. Bold: best score, Underline: second score. In brackets: estimated standard deviation.

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	<u>14.48</u>(0.37)	**2.88**(0.05)	3.986(0.93)	1561.14(21.61)
Trunc. exch.	**14.42**(0.37)	<u>2.94</u>(0.05)	3.989(0.93)	1561.16(21.69)
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)
Algo 8 ($m = 2$)	8.25(0.21)	2.57(0.05)	3.987(0.93)	1561.16(21.62)

Table 5: S&P 500 $n = 2023$, $L_D = 200$, $L_M = 500$. Bold: best score, Underline: second score. In brackets: estimated standard deviation.

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	<u>22.58</u>(0.75)	**145.07**(3.06)	4.977(0.82)	14990.47(57.56)
Trunc. exch.	**21.59**(0.72)	<u>148.66</u>(3.14)	4.978(0.82)	14990.50(59.09)
Truncated	32.75(1.09)	148.69(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)
Algo 8 ($m = 2$)	13.55(0.45)	106.28(2.24)	4.980(0.82)	14990.38(59.09)

Table 6: Bimod data $n = 1000$, $L_D = 150$, $L_M = 800$. Bold: best score, Underline: second score. In brackets: estimated standard deviation.

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	<u>93.28</u>(3.93)	**3.65**(0.07)	3.806(1.73)	2735.14(8.66)
Trunc. exch.	**91.20**(3.85)	<u>3.69</u>(0.07)	3.795(1.72)	2735.14(8.66)
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)
Algo 8 ($m = 2$)	47.25(1.99)	3.06(0.06)	3.798(1.72)	2735.14(8.65)

Table 7: Lepto data $n = 1000$, $L_D = 150$, $L_M = 800$. Bold: best score, Underline: second score. In brackets: estimated standard deviation.

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	**235.49**(9.93)	<u>13.17</u>(0.24)	4.006(2.05)	2400.51(18.68)
Trunc. exch.	<u>237.70</u>(10.02)	13.66(0.25)	4.022(2.08)	2400.48(18.72)
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)
Algo 8 ($m = 2$)	148.81(6.28)	11.55(0.21)	4.018(2.07)	2400.48(18.69)

Table 8: Bimod data $n = 10,000$, $L_D = 200$, $L_M = 1000$. Bold: best score, Underline: second score. In brackets: estimated standard deviation.

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	**244.64**(11.53)	5.79(0.12)	3.77(1.74)	27235.46(9.09)
Trunc. exch.	<u>247.02</u>(11.65)	<u>5.57</u>(0.12)	3.77(1.73)	27235.45(9.04)
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)
Algo 8 ($m = 2$)	180.58(8.51)	4.76(0.10)	3.78(1.76)	27235.46(9.05)

Table 9: Lepto data $n = 10,000$, $L_D = 200$, $L_M = 1000$. Bold: best score, Underline: second score. In brackets: estimated standard deviation.

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	212.65(10.03)	<u>11.74</u>(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)
Truncated	<u>186.83</u>(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)
Algo 8 ($m = 2$)	142.67(6.73)	10.47(0.22)	3.74(1.77)	23517.95(12.47)

PYM case

We now provide the results obtained in the PYM case with $d = 0.3$ and $\alpha = 1$, only for Galaxy and lepto data with $n = 1,000$. These results are given in Tables 10-11. The rest of the results are detailed in Appendix C.

Table 10: Galaxy data $n = 82$, $L_D = 150$, $L_M = 300$. Bold: best score, Underline: second score. In brackets: estimated standard deviation.

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	<u>10.56</u>(0.27)	<u>2.84</u>(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)
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)
Algo 8 ($m = 2$)	5.79(0.15)	2.37(0.04)	4.869(2.13)	1561.66(21.89)

Table 11: Lepto data $n = 1000$, $L_D = 200$, $L_M = 1000$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	**156.21**(7.37)	<u>13.56</u>(0.29)	4.255(3.22)	2371.35(16.11)
Trunc. exch.	<u>167.13</u>(7.88)	**13.08**(0.28)	4.247(3.13)	2371.34(16.03)
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)
Algo 8 ($m = 2$)	96.90(4.57)	10.90(0.23)	4.242(3.22)	2371.34(15.98)

The results presented here and in the appendices show that, in almost all situations, the two variants of the proposed method outperform the other competitors in conditional algorithms, thanks to the exchangeability in the model and the introduction of the proposed threshold. The "Slice efficient" gives the worst performance.

To better understand and explain these results, we now investigate the gain in mixing performance of the algorithms due to the exchangeability property of the model, on the one hand, and the proposed threshold, on the other. For this reason, we implement our slice sampler using the exchangeable model but without the threshold ("SE without thres."), and the "Slice efficient" of [26] which uses a non-exchangeable model with the introduction of our threshold ("Slice eff. thres."). Tables 12-13 show the results on the real data (Galaxy and S& P 500). This work has been carried out on all the datasets considered, but for reasons of space we present only the results obtained on these datasets.

Firstly, as far as the threshold is concerned, its inclusion in the "Slice efficient" significantly reduces the IAT, as can be seen by comparing the original "Slice efficient" and its thresholded version "Slice eff. thres.". The performance of the latter is close to that of the "Truncated". We believe that the poor mixing due to non-exchangeability in the stick-breaking posterior representation is accentuated by the absence of weights in the "Slice efficient". This could often hinder the Gibbs sampler in the allocation step, moving an observation from one component associated with a few observations to another associated with many observations. The introduction of our threshold facilitates this change. We remind that the "Truncated" algorithm takes into account the weights of the mixture components when updating the classification variables. On the flip side, removing the threshold in our sampler increases the IAT. This is noticeable

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

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # 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 13: S&P 500 $n = 2023$, $L_D = 200$, $L_M = 500$.

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # 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)

in the differences between "Slice exch. thres" and "SE. without thres.". Overall, it was observed for all datasets that the introduction of our threshold leads to a faster decrease in autocorrelation curves in the early lags, and therefore a lower IAT.

We now look at the benefits we derive from the model's exchangeability property. This is reflected in the differences between "Slice exch. thres." and "Slice eff. thres.", and between "Trunc. exch." and "Truncated". Here too, exchangeability reduces IAT. We also noted that the autocorrelation curves obtained by "Slice exch. without thres." decrease and tend towards zero more rapidly than in the algorithms using non-exchangeable models ("Truncated", "Slice eff. thres." and "Slice efficient"). This behaviour was observed for all datasets.

Finally, it is important to point out that the two proposed variants and Algorithm 8 of [19] were stable in all experiments: for various simulations, we always obtained the same results for each data set and data size. In contrast, algorithms using non-exchangeable models ([20] and [26]) did not always give the same results. We also observed erratic convergence behaviour of the Gibbs sampler in these two algorithms, particularly for large datasets (e.g. lepto with $n = 10,000$).

4 Conclusion and discussion

When models become increasingly complex due to increasing dimensionality, the poor mixing of an MCMC algorithm can be particularly inhibiting. It then seems important to develop samplers that improve algorithm mixing while experimenting with strategies to reduce computational cost. In this paper, we have proposed a simple, efficient

and easy-to-use Gibbs sampler for posterior simulation under Pitman-Yor and Dirichlet mixture models, which meets the constraint of efficient parallelization capability while retaining good mixing properties. Our comparative study on real and simulated data sets confirm our belief that the two variants of our conditional Gibbs sampler have the potential to be a useful and interesting addition to the menu of samplers for DPM and PYM. A difference between the two proposed variants is that for the truncated version ("Trunc. exch."), the fixed length of the approximation must be decided before the sampling. Most of the time, this is not a crucial point for Dirichlet process mixtures, but for the two-parameter case, the fixed approximation may give rise to biased estimates for moderate truncation lengths. The exchangeable thresholded slice version ("Slice exch. thres.") achieves adaptive truncation at each iteration, and maintains a nice trade-off between IAT and time cost.

Our samplers have been developed for Pitman-Yor and Dirichlet process mixture models. Since the "Slice efficient" of [26] has been developed for more general stick-breaking priors, and the introduction of our proposed threshold has been shown to improve its mixing property, this may be an interesting solution for more general stick-breaking processes other than DP and PYP. In this case, an attractive perspective might be to also introduce mixing moves over clusters labels as suggested by [29] and [24]. As mentioned, the order of clusters labels matters in the stick-breaking representation. A label permutation step could result in a better mixing chain.

Appendix A

This appendix presents the rest of the results for the DPM case, with $\alpha = 1$ These are the tables for Lepto and Bimod simulated data with $n = 100$. We also present some figures relating to the autocorrelation curves and densities estimated by competing algorithms.

Table A1: Bimod data $n = 100$, $L_D = 150$, $L_M = 300$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	<u>28.76</u>(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)
Truncated	54.38(1.40)	<u>5.89</u>(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)
Algo 8 ($m = 2$)	15.59(0.40)	5.20(0.09)	3.794(1.66)	287.59(8.52)

Table A2: Lepto data $n = 100$, $L_D = 200$, $L_M = 500$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	<u>25.61</u>(0.85)	<u>13.53</u>(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)
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)
Algo 8 ($m = 2$)	14.79(0.49)	9.83(0.28)	3.994(1.63)	239.72(11.36)

Figures A1 and A2 show, for Galaxy data, the autocorrelation curves that were used to estimate IAT on the number of clusters and on deviance respectively. Figure A3 displays the data histogram and estimated posterior density for each algorithm.

Fig. A1: Autocorrelation curves used to estimate the IAT for the number of clusters (Galaxy data).

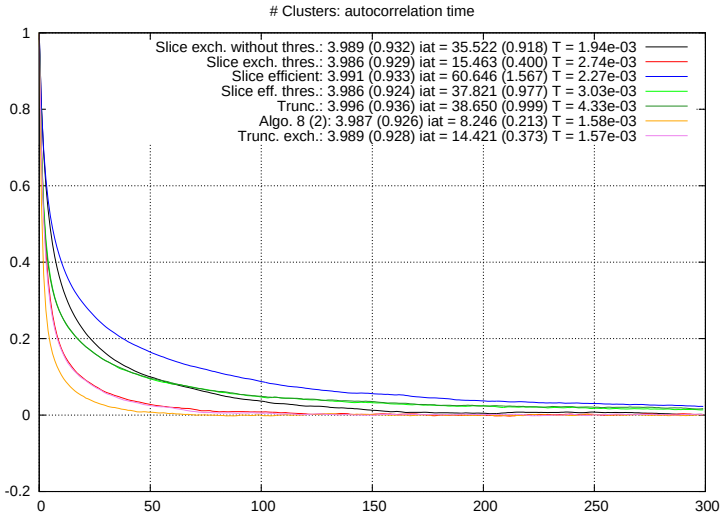


Fig. A2: Autocorrelation curves used to estimate the IAT for the deviance (Galaxy data).

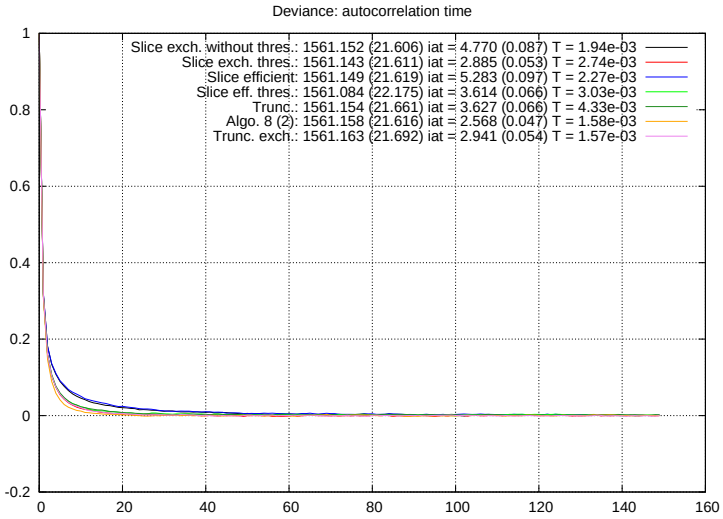
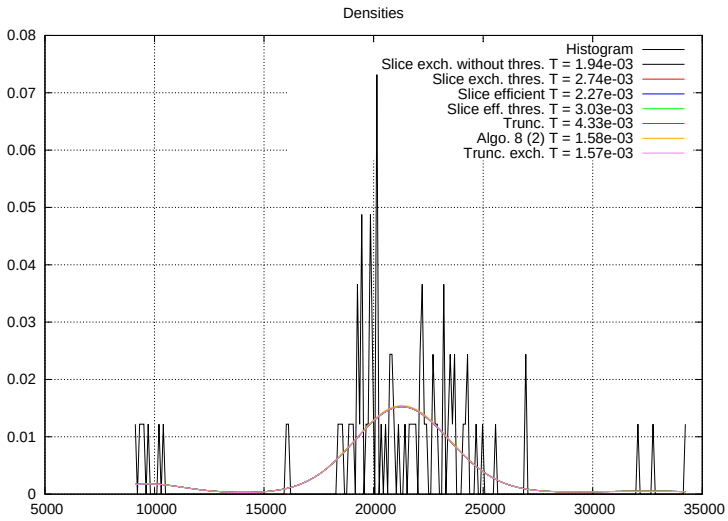


Fig. A3: Histogram of data and estimated densities (Galaxy data).



Appendix B

This appendix presents the results for the DPM case, with $\alpha = 5$, with the exception of the results for lepto and bimod data with n =10,000.

Table B3: Galaxy data $n = 82$, $L_D = 150$, $L_M = 300$. Bold: best score, Underline: second score. In brackets: estimated standard deviation.

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	<u>10.73</u>(0.28)	**2.80**(0.05)	7.082(3.32)	1563.10(23.55)
Trunc. exch.	**10.11**(0.26)	<u>2.81</u>(0.05)	7.084(3.31)	1563.10(23.54)
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)
Algo 8 ($m = 2$)	6.16(0.16)	2.35(0.04)	7.084(3.31)	1563.10(23.56)

Table B4: Bimod data $n = 100$, $L_D = 150$, $L_M = 300$. Bold: best score, Underline: second score. In brackets: estimated standard deviation.

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	<u>14.24</u>(0.37)	<u>2.35</u>(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)
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)
Algo 8 ($m = 2$)	8.56(0.22)	2.01(0.04)	8.888(5.42)	283.18(8.98)

Table B5: Bimod data $n = 1000$, $L_D = 150$, $L_M = 800$. Bold: best score, Underline: second score. In brackets: estimated standard deviation.

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	<u>81.36</u>(3.43)	<u>6.90</u>(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)
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)
Algo 8 ($m = 2$)	42.85(1.81)	5.35(0.10)	9.928(7.36)	2741.08(12.03)

Appendix C Appendix 3

This appendix presents the results for the PYM case, with $d = 0.3$ and $\alpha = 1$, excluding Galaxy and lepto 1000, already presented in the main paper.

Table B6: Lepto data $n = 100$, $L_D = 200$, $L_M = 500$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	**11.12**(0.37)	**14.26**(0.30)	9.004(4.74)	257.17(21.70)
Trunc. exch.	<u>11.84</u>(0.39)	<u>14.34</u>(0.30)	8.988(4.75)	257.19(21.89)
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)
Algo 8 ($m = 2$)	6.95(0.23)	11.43(0.24)	8.999(4.73)	257.18(21.66)

Table B7: Lepto data $n = 1000$, $L_D = 150$, $L_M = 800$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	<u>90.94</u>(3.84)	<u>15.81</u>(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)
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)
Algo 8 ($m = 2$)	50.75(2.14)	13.13(0.24)	11.098(7.69)	2354.94(19.48)

Table B8: S&P 500 data $n = 2023$, $L_D = 300$, $L_M = 500$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	**17.09**(0.57)	151.34(3.91)	7.476(2.65)	14989.81(53.22)
Trunc. exch.	<u>18.02</u>(0.60)	**143.45**(3.71)	7.484(2.65)	14989.68(51.86)
Truncated	21.73(0.72)	<u>150.94</u>(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)
Algo 8 ($m = 2$)	11.33(0.38)	97.28(2.51)	7.478(2.65)	14989.76(52.48)

Table C9: Bimod data $n = 100$, $L_D = 150$, $L_M = 300$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	**25.77**(0.67)	**7.75**(0.14)	4.726(3.42)	267.96(9.97)
Trunc. exch.	<u>26.47</u>(0.68)	<u>7.97</u>(0.15)	4.650(3.06)	267.93(9.99)
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)
Algo 8 ($m = 2$)	14.27(0.37)	5.79(0.11)	4.720(3.40)	267.95(9.99)

Table C10: Bimod data $n = 1000$, $L_D = 150$, $L_M = 800$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	**42.74**(1.80)	<u>2.60</u>(0.05)	4.427(3.21)	2646.88(9.02)
Trunc. exch.	<u>46.05</u>(1.94)	**2.54**(0.05)	4.401(3.05)	2646.88(9.01)
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)
Algo 8 ($m = 2$)	22.80(0.96)	2.15(0.04)	4.425(3.18)	2646.88(8.99)

Table C11: Lepto data $n = 100$, $L_D = 200$, $L_M = 500$. Bold: best score, Underline: second score. In brackets: estimated standard deviation

	IAT on # of clusters ↓	*IAT on deviance* ↓	*Estimated # clusters*	*Estimated deviance*
Slice exch. thres	**47.85**(1.60)	<u>27.00</u>(0.57)	3.719(3.70)	223.92(8.55)
Trunc. exch.	<u>50.04</u>(1.67)	**26.91**(0.57)	3.674(3.39)	223.86(8.61)
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)
Algo 8 ($m = 2$)	27.28(0.91)	17.83(0.38)	3.720(3.71)	223.92(8.55)

References

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

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

[3] Müller, P., Quintana, F.A., Jara, A., Hanson, T.: Bayesian Nonparametric Data Analysis. Springer Series in Statistics. Springer, (2015).

[4] Ghosal, S., Vaart, A.W.: Fundamentals of Nonparametric Bayesian Inference. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, (2017).

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

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

[7] Favaro, S., Walker, S.G.: Slice sampling σ-stable Poisson-Kingman mixture models. Journal of Computational and Graphical Statistics **22**, 830–847 (2013)

[8] Nieto-Barajas, L.E., Prünster, I., Walker, S.G.: Normalized random measures driven by increasing additive processes. The Annals of Statistics **32**(6), 2343–2360 (2004) https://doi.org/10.1214/009053604000000625

[9] Favaro, S., Teh, Y.W.: MCMC for normalized random measure mixture models. Statistical Science **28**(3) (2013) https://doi.org/10.1214/13-sts422

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

[11] Neal, R.M.: 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, pp. 197–211 (1991)

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

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

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

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

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

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

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

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

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

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

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

[23] Arbel, J., De Blasi, P., Prünster, I.: Stochastic approximations to the Pitman–Yor process. Bayesian Analysis **14**(4) (2019) https://doi.org/10.1214/18-ba1127

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

[25] Papaspiliopoulos, O.: A note on posterior sampling from Dirichlet mixture models. Preprint (2008)

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

[27] Gelfand, A.E., Kottas, A.: A computational approach for full nonparametric Bayesian inference under Dirichlet process mixture models. Journal of Computational and Graphical Statistics **11**(2), 289–305 (2002)

[28] Arbel, J., Lijoi, A., Nipoti, B.: Full Bayesian inference with hazard mixture models. Computational Statistics & Data Analysis **93**, 359–372 (2016)

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

[30] Lomeli, M., Favaro, S., Teh, Y.W.: A hybrid sampler for Poisson-Kingman mixture models. Advances in Neural Information Processing Systems **28** (2015)

[31] Dubey, K.A., Zhang, M., Xing, E., Williamson, S.: Distributed, partially collapsed MCMC for Bayesian nonparametrics. In: International Conference on Artificial Intelligence and Statistics, pp. 3685–3695 (2020). PMLR

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

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

[34] Caron, F., Teh, W., Yee, M., Brendan, T.: Bayesian nonparametric Plackett-Luce models for the analysis of preferences for college. Annals of Applied Statistics, 8 (2): 1145-1181 (2014)

[35] Donnet, S., Rivoirard, V., Rousseau, J., Scricciolo, C.: Posterior concentration rates for empirical Bayes procedures with applications to Dirichlet process mixtures. Bernoulli **24**(1), 231–256 (2018). arXiv: 1406.4406v1

[36] Canale, A., Corradin, R., Nipoti, B.: Importance conditional sampling for pitman–yor mixtures. Statistics and Computing **32**(3), 40 (2022)

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

[38] Pitman, J.: Some developments of the Blackwell-MacQueen urn scheme. In: al., T.S.F. (ed.) Statistics, Probability and Game Theory; Papers in Honor of David Blackwell. Lecture Notes-Monograph Series, vol. 30, pp. 245–267. Institute of Mathematical Statistics, (1996)

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

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